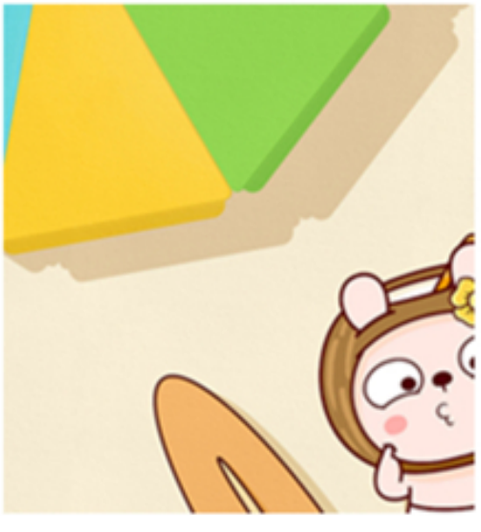
最新IT书籍PDF
最新IT书籍PDF
最新IT书籍PDF
最新IT书籍PDF



版权相关注意事项：
1、书籍版权归著者和出版社所有
2、本PDF来自于各个广泛的信息平台，经过整理而成
3、本PDF仅限用于非商业用途或者个人交流研究学习使用
4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

版权相关注意事项：
1、书籍版权归著者和出版社所有
2、本PDF来自于各个广泛的信息平台，经过整理而成
3、本PDF仅限用于非商业用途或者个人交流研究学习使用
4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

非卖品，仅供非商业用途或交流学习使用

CETC 云计算技术与应用专业校企合作系列教材

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

视频直播APP应用开发

版权所有，严禁传播，违者自负法律责任！

主　编　陆碧霞　邱跃鹏
副主编　李筱林　向春枝　喻　洁
主　审　武海华　罗主山

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！

高等教育出版社

版权所有，严禁传播，违者自负法律责任！
非卖品，仅供非商业用途或交流学习使用

CETC 云计算技术与应用专业校企合作系列教材

视频直播APP应用开发

主　编　眭碧霞　邱跃鹏
副主编　李筱林　向春枝　喻　洁
主　审　武海华　罗宝山

高等教育出版社·北京

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

内容简介

本书是云计算技术与应用专业校企合作系列教材。

本书较为全面地介绍了目前流行的视频直播相关知识，依托腾讯云直播服务平台，介绍了腾讯云直播（Live Video Broadcasting）快速的直播接入和分发服务。同时，本书以“斗学直播”APP 为项目原型，以任务驱动为主线，基于腾讯云平台，完成了腾讯云直播服务的申请与购买、成功案例的体验、直播 SDK 的介绍、“斗学直播”APP 的完整案例开发等内容，并全面翔实地讲述了推流技术的相关知识。

整个“斗学直播”项目包含视频直播基础调研、功能设计和服务租赁、直播搭建开发环境、用户管理模块、直播模块、直播互动模块、直播后台管理及应用发布这 8 部分。每部分都包含若干分解任务，每个分解任务包含其相关的基本知识学习和步骤实现两部分。在阐述中，本书尽量做到基础知识介绍具有针对性，任务目标操作具体化。每个任务结束设计有拓展实训供读者练习和提高。最后的附录提供了 iOS 直播、小程序直播开放的案例，供读者学习和参考。通过本书的学习可以让读者更加全面地了解 Android 组件的应用场景及方式，以及 Web 和 Android 前后台交互实现的方式。

本书可以作为高职高专院校软件技术专业和计算机网络技术专业的基础核心课程以及计算机相关专业的移动应用开发课程的教材，也可以作为软件技术应用和移动应用开发技术入门的培训教材，同时适合腾讯云开发专业人员、直播爱好者和广大计算机爱好者自学使用。

图书在版编目（CIP）数据

视频直播 APP 应用开发 / 眭碧霞，邱跃鹏主编. -- 北京：高等教育出版社，2018.1

ISBN 978-7-04-049016-9

Ⅰ. ①视… Ⅱ. ①眭… ②邱… Ⅲ. ①移动终端-应用程序-程序设计-高等职业教育-教材 Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字（2017）第 293696 号

策划编辑	许兴瑜	责任编辑	许兴瑜	封面设计	赵 阳	版式设计	徐艳妮
插图绘制	邓 超	责任校对	刘 莉	责任印制	毛斯璐		

出版发行	高等教育出版社	网 址	http://www.hep.edu.cn
社 址	北京市西城区德外大街 4 号		http://www.hep.com.cn
邮政编码	100120	网上订购	http://www.hepmall.com.cn
印 刷	高教社（天津）印务有限公司		http://www.hepmall.com
开 本	787mm × 1092mm 1/16		http://www.hepmall.cn
印 张	15.25		
字 数	370 千字	版 次	2018 年 1 月第 1 版
购书热线	010-58581118	印 次	2018 年 1 月第 1 次印刷
咨询电话	400-810-0598	定 价	38.50 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换

版权所有　侵权必究
物 料 号　49016-00

版权所有，严禁传播，违者自负法律责任！

版权相关注意事项：
1、书籍版权归著者和出版社所有
2、本PDF来自于各个广泛的信息平台，经过整理而成
3、本PDF仅限用于非商业用途或者个人交流研究学习使用
4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

非卖品，仅供非商业用途或交流学习使用

前　言

一、缘起

腾讯云是腾讯公司倾力打造的面向广大企业和个人的公有云平台，除了提供云服务器、云数据库、云存储、安全、大数据、CDN 等基础云计算服务外，还提供游戏、视频直播、移动应用、VR 视频、物联网、大数据、人工智能等整套打包解决方案。从 2010 年上线提供服务起，腾讯云逐渐成为全球最具有特色的公有云服务平台。

- 腾讯云为社交化的微信开放平台提供基础平台服务，为客户和创业者构建互联的应用生态。
- 腾讯云游戏解决方案是行业标杆客户的第一选择，为游戏行业用户量身定制，借助腾讯云实现游戏的快速发展，轻松运维。
- 腾讯云直播视频提供了完整的视频点播、直播、互动直播和云通信一站式服务，用户可快速搭建一个功能齐备的游戏直播发布平台。
- 腾讯云推出以场景应用落地为目标的人工智能 AI 服务，引领 AI 创新创业。
- 腾讯云联合投资机构、孵化器和产业园区等推出的创业扶持计划“云+创业”计划。为创业者提供全方位服务，帮助创业者轻松上云，快速腾飞。
- 腾讯云的“云+校园”计划，为高校生提供先进的技术支持、资金扶持和经验分享，更多高校生学习云计算及互联网知识，为后续职业、创业发展奠定基础。

腾讯云的定位以连接为主，自腾讯云全面开放以来，很多成功的互联网企业都在采用腾讯云。

二、结构

本教材教学内容同时采用模块化的编写思路，由浅入深，以开发一个视频直播应用为主线，贯彻产品需求、服务构建、APP 开发、后台 Web 开发和产品发布上云的整个实际应用项目。

每个单元通过学习情境引出单元的教学核心内容，明确教学任务。每个任务的编写分为任务描述、知识学习、任务实现和项目实训 4 个环节。

- 任务描述：简述任务目标，展示任务实施效果，提高学生学习兴趣；
- 知识学习：详细讲解知识点，通过系列实例实践，边学边做；
- 任务实现：通过任务综合应用所学知识，提高学生系统地运用知识的能力；
- 项目实训：在项目实施的基础上通过“学、仿、做”达到理论与实践统一、知识内化的教学目的。

最后进行单元小结，总结本单元的教学重点、难点。

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

三、特点

1. 针对性强，教材内容的选取以实用为主

本教材以云计算技术专业学生的就业岗位群为导向，整个课程为两个部分：环境准备与项目实现。

环境准备涉及腾讯云服务的申请与开发环境的搭建配置。项目实现以正常开发顺序完成整个直播应用项目。

2. 精心设计，教学内容与数字化资源有机结合

本教材以教学内容为主线，将各项数字化资源有机地结合在一起，形成完整的数字课程。

数字化资源包括 3 个方面的内容：第一，课程本身的基本信息，包括课程简介、学习指南、课程标准、整

体设计、单元设计、考核方式等；第二，教学内容中重点、难点的微课视频教学资源，既方便课内教学，又方便学生课外预习和复习；第三，课程拓展资源，包括课程的重点、难点剖析，循序渐进的综合项目开发、相关培训、认证、案例、素材资源等。

教材内容满足课堂教学的需要，而数字化资源为学生课外自主探究学习提供了一个良好的平台，课堂教学与智慧职教平台结合，提高了教学效果与学习效果。

四、使用

1. 教学内容课时安排

本教材建议授课 64 学时，教学单元与课时安排如表 1 所示。

表 1　教学单元与课时安排

序号	单元名称	学时安排
1	视频直播基础调研	4
2	功能设计和服务租赁	4
3	搭建开发环境	8
4	用户管理模块	10
5	直播模块	14
6	直播互动模块	8
7	直播后台管理	12
8	应用发布	4
课时总计		64

2. 课程教学资源一览表

本教材是云计算技术与应用专业校企合作系列教材，开发了丰富的数字化教学资源，可使用的教学资源如表 2 所示。

表 2　课程教学资源一览表

序号	资源名称	表现形式与内涵
1	课程简介	Word 文档，包括对课程内容的简单介绍和对课时、适用对象等项目的介绍，让学生对直播应用有一个简单的认识
2	学习指南	Word 文档，包括对学前要求、学习目标要求及学习路径和考核标准要求，让学生知道如何使用资源完成学习

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

续表

序号	资源名称	表现形式与内涵
3	课程标准	Word 文档，包括课程定位、课程目标要求及课程内容与要求，可供教师备课时使用
4	整体设计	Word 文档，包括课程设计思路、课程具体的目标要求，以及课程内容设计和能力训练设计，同时给出考核方案设计，让教师理解课程的设计理念，有助于教学实施
5	授课 PPT 和视频	PPT 文件和 MP4 视频文件，可帮助教师理解如何教好视频直播 APP 应用开发这门课
6	教学单元设计	Word 文档，分任务给出课程教案，帮助教师完成一堂课的教学细节分析
7	微课	MP4 视频文件，使学生更加直观地学习，有助于学习知识
8	电子课件	PPT 文件，提供 PowerPoint 2010 版的使用，也可以根据教师实际需要加以修改后使用
9	实训任务单	Word 文档，为每个任务设计实训来加深课堂知识的学习，并给出实训的详细实训步骤
10	案例	Tar 包，包括单元项目案例和综合案例，综合运用所学的知识
11	习题库、试卷库	Word 文档，习题包括理论习题和操作习题，试卷包括单元测试和课程测试。通过练习和测试，让学习者加深对知识的掌握程度
12	附书源码	Tar 包，包括书中所有例题和任务的源代码

本书配套的资源包、运行脚本、电子教案等，读者可登录 http://www.1daoyun.com 下载。

五、致谢

本书由腾讯云、南京第五十五所技术开发有限公司和江苏一道云科技发展有限公司共同编写，由高等教育出版社出版。

由于作者水平有限，错误和不足之处在所难免，恳请各位读者给予批评、指正，编者将不胜感激。

编　者

2017 年 12 月

版权所有，严禁传播，违者自负法律责任！

版权相关注意事项：
1、书籍版权归著者和出版社所有
2、本PDF来自于各个广泛的信息平台，经过整理而成
3、本PDF仅限用于非商业用途或者个人交流研究学习使用
4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

非卖品，仅供非商业用途或交流学习使用

目　录

单元 1　视频直播基础调研　1

学习目标　1
学习情境　2
任务 1.1　视频直播市场的初步调研　2
任务描述　2
知识学习　2
任务实现　3
项目实训　5
任务 1.2　GoGal 小直播技术调研　6
任务描述　6
知识学习　6
任务实现　6
项目实训　9
单元小结　9

单元 2　功能设计和服务租赁　11

学习目标　11
学习情境　12
任务 2.1　功能设计和技术选型　12
任务描述　12
知识学习　12
任务实现　22
项目实训　33
任务 2.2　服务租赁　33
任务描述　33
知识学习　34
任务实现　60
项目实训　69
单元小结　70

单元 3　搭建开发环境　71

学习目标　71
学习情境　72
任务 3.1　了解 RTMP SDK 并下载　72
任务描述　72
知识学习　72
任务实现　73
项目实训　75
任务 3.2　安装并配置开发环境　75
任务描述　75
知识学习　75
任务实现　77
项目实训　94
单元小结　94

单元 4　用户管理模块　95

学习目标　95
学习情境　96
任务 4.1　实现用户注册模块　96
任务描述　96
知识学习　96
任务实现　96
项目实训　104
任务 4.2　实现用户登录模块　104
任务描述　104

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

知识学习 104
任务实现 106
项目实训 110
任务 4.3 实现个人中心模块 110
任务描述 110
知识学习 111
任务实现 112
项目实训 118
单元小结 118

单元 5 直播模块 119

学习目标 119
学习情境 120
任务 5.1 实现开启直播模块 120
任务描述 120
知识学习 120
任务实现 121
项目实训 145
任务 5.2 实现直播列表模块 145
任务描述 145
知识学习 146
任务实现 147
项目实训 154
任务 5.3 实现观看直播模块 155
任务描述 155
知识学习 155
任务实现 155
项目实训 161
单元小结 161

单元 6 直播互动模块 163

学习目标 163
学习情境 164
任务 6.1 实现点赞模块 164
任务描述 164
知识学习 164
任务实现 166
项目实训 168
任务 6.2 实现弹幕模块 168
任务描述 168
知识学习 168
任务实现 169
项目实训 174
任务 6.3 实现功能扩展 174
任务描述 174
知识学习 175
任务实现 175
项目实训 177
单元小结 177

单元 7 直播后台管理 179

学习目标 179
学习情境 180
任务 7.1 实现登录注销功能 180
任务描述 180
知识学习 180
任务实现 180
项目实训 185
任务 7.2 实现直播管理功能 185
任务描述 185
知识学习 185
任务实现 186
项目实训 201
任务 7.3 实现用户管理功能 201
任务描述 201
知识学习 201
任务实现 202
项目实训 213
单元小结 213

单元 8 应用发布 215

学习目标 215
学习情境 216
任务 8.1 部署 Web 项目 216

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

任务描述 216
知识学习 216
任务实现 217
项目实训 222
任务 8.2 签名打包 223
任务描述 223
知识学习 223
任务实现 225
项目实训 230
单元小结 230
附录 iOS 和小程序直播 231
参考文献 232

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

单元 1 视频直播基础调研

学习目标

PPT1：
视频直播基础

知识目标

- 了解视频直播的基本概念。
- 了解视频直播类型。
- 了解视频直播原理。

技能目标

- 体验基于腾讯云服务的成功案例。
- 体验腾讯云直播。

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

学习情境

从事教育行业的某公司想开发一款视频直播 APP，需要能够满足用户在 APP 客户端开启视频直播和观看视频直播。主播和观众能够进行一些简单的互动，比如弹幕、点赞等，支持主播使用美颜、背景音乐等直播辅助功能。但是公司对直播 APP 开发不是很了解，需要对其基本内容和架构原理进行简单的调研。公司将这些需求交给了研发部的工程师小赵，于是小赵对此展开了一系列的调研工作。

任务 1.1　视频直播市场的初步调研

微课 1-1
视频直播基础

任务描述

调查视频直播的基本概念、常见类型、基本原理及其基础架构，研究视频直播的基本业务流程，调研并体验腾讯云直播的一些成功案例。

知识学习

1. 视频直播概念

视频直播，泛指网络视频直播，是指利用互联网及流媒体技术将融合图像、文字、声音等丰富元素的视频，通过直播的形式实时展现出来，进而达到声形并茂的效果。

2. 视频直播的主要功能

包括直播间的创建和关闭、直播互动、消息发送、录播点播、实时统计、实时监控、分享等。

3. 视频直播技术

视频直播技术涉及音视频采集技术、前处理、编解码、多媒体处理框架、音视频 Mux/Demu（多路传输、混流）、流媒体协议、数据库、缓存技术、分布式文件系统、CDN 技术、播放控制、交互技术、水印弹幕等。

4. 视频直播的表现形式

移动视频直播主要指用户通过移动端进行制作或观看直播视频，其中包括制作者通过 PC 端进行视频直播，而观看者通过移动端进行观看；或制作者通过移动端进行视频直播，而观看者通过 PC 端或移动端进行观看等。

主要表现形式大多为移动视频直播平台在移动端 APP 为用户提供服务。

5. 视频直播的常见类型

常见直播 APP 如图 1-1 所示。

（1）传统秀场类直播

秀场类直播是音乐经济与美女经济的结合，内容主要是唱歌、跳舞、才艺表演。国内的秀场类直播平台主要有 YY、9158、六房间、百家百秀、网易 BoBo、新浪秀场等。

图 1-1
直播 APP

（2）游戏类直播

游戏类直播内容主要是以主播玩游戏为主，它既包括网络游戏直播、单机游戏直播、其他游戏直播，也包括户外娱乐直播等游戏娱乐内容。

（3）泛娱乐类直播

泛娱乐直播并不专注于单一直播领域，而是面向所有可直播的领域，如演唱会、户外、御宅等，趁着移动互联网发展成熟、获得便捷，形成了全民直播之势。包括映客、花椒等移动直播平台，YY、斗鱼、虎牙等跨平台直播平台。

（4）其他垂直类直播

垂直类直播是与全民直播相区别的直播类型，它通常围绕单一主题，在用户内容的基础上配以一些专业内容。除了游戏直播外，垂直类直播还涉及商务领域、教育领域、财经领域等。

（5）VR 直播

随着脸书（Facebook）、索尼（Sony）等公司在虚拟现实技术（Virtual Reality，VR）硬件技术上的进步，VR 技术逐渐从幕后走向台前，VR 与视频直播的结合可能将成为视频直播领域的下一个风口。

任务实现

1. 案例分析 1：斗鱼 TV 全民直播平台

小赵通过登录斗鱼网站进行视频直播体验、搜索互联网、翻阅书籍和论文、请教行业前辈，终于完成了调研的基本工作，并整理成笔记，内容如下。

（1）客户简介

斗鱼全民游戏直播平台提供高清、快捷、流畅的视频直播和游戏赛事直播服务，包含英雄联盟（LOL）直播、穿越火线（CF）直播、Dota2 直播、美女直播等各类热门游戏赛事直播和各家直播活动。活动直播频道约有 8 000 多个，同时在线的观众有数百万，是国内目前最大的游戏竞技类直播平台。

（2）业务需求

斗鱼 TV 面对覆盖全国的庞大用户群体，以及游戏直播领域的新兴竞争对手，急需提升直播质量，通过更为清晰、无卡顿的核心能力服务用户。此外，在传统直播功能之外还

笔 记

需要完善其他功能，包括录制、截图、鉴黄等，实现全面的直播管理。

另外，传统的直播构架存在 FLV/RTMP 多协议下发管理难度大、统计不准确等各种问题，这些都需要一并解决。尤其是统计功能，及时、准确和多维度的统计报表，对于提高业务运维策略，将会非常重要。

同时，由于直播业务的特殊性，客户深受涉黄视频内容的困扰。投入了大量的人力，采用半人工的方式进行鉴黄工作，成本高，效果不好。

（3）解决方案

通过深入沟通，腾讯云团队对斗鱼的直播核心业务进行了详细的构架梳理，帮助斗鱼 TV 打造了新一代直播核心功能。

（4）核心功能

- 支持主播端推两路直播流相互备份，提供高可靠的直播推流方案。
- 支持直播多码率转码，便于客户实现适应于不同播放场景的下发方案，并且极大地节省了带宽成本。
- 支持 RTMP/FLV 协议从第三方直播平台回源，以提供灵活的直播调度功能。
- CDN 直播加速支持 FLV（1 路主线路）和 RTMP（3 路备用线路），具备高可用直播流。
- 提供统计模块，支持实时在线用户信息收集统计。
- 日志模块支持与斗鱼管理平台对接，实现日志推送，便于通过灵活的日志格式洞察业务细节。

（5）增值服务

- 提供录播功能，支持直播回看功能，实现直播内容的增值使用。
- 结合行业领先的腾讯云优图图片分析，对直播内容过程进行截图识别，减轻人工审核负担。

（6）方案效果

通过改造，斗鱼 TV 凭借腾讯云提供的直播构架，在核心能力——包括直播质量、清晰度、流畅度和统计等功能上，完全满足了遍布全国的客户的观看要求。

在近期举行的持续 LOL、Dota2 等直播活动中，轻松应对了在线人数 10 万以上的苛刻要求，以清晰的画质处理和强大的资源调度，帮助覆盖全国的用户获得高质量的流畅直播观看体验。

2. 案例分析 2：快手短视频 APP

小赵通过快手客户端进行视频直播体验、搜索互联网、翻阅书籍和论文、请教行业前辈，终于完成了调研的基本工作，并整理成笔记，内容如下：

（1）客户简介

快手是一款于 2011 年诞生的短视频应用，用户可以通过它制作并分享短视频。目前在全球超过 5 亿用户使用快手，每天数百万新鲜的原创视频在快手诞生。腾讯云作为快手的战略合作伙伴，为快手提供全面的云端视频服务。

（2）业务需求

快手在全球拥有超过 5 亿用户，每日产生数百万原创短视频，需要保证全球范围视频的播放流畅，存储完整。

- 希望为用户提供差异化的互动直播服务。
- 弹性转码需求量：自定义转码需求，应对不同的应用场景。
- 特殊需求的通信服务：上行短信需求与用户登录验证短信（多语言）。
- 人工审核与运营成本繁重，希望可以使用智能 AI 来提高运营效率。

（3）腾讯云解决方案

腾讯云在全球有 600+的加速节点，为快手提供稳定流畅的直播点播加速服务，同时为快手提供稳定高效的云端存储服务，很好地支持了快手海量视频存储，以及直播点播的业务高速发展。

腾讯云为快手提供了全新升级的互动直播 SDK，支持在 1 天之内快速实现音/视频采集、动效、播放、连麦、通话等场景，最多支持 10 人的视频连麦，同时支持跨房连麦功能。

腾讯视频云基于腾讯长期在音视频领域的技术积累和能力，拥有超大规模 12000+的分布式转码集群，为快手按需提供转码服务，最大限度地满足快手的转码需求，避免资源浪费。腾讯视频云转码为快手提供了强大的视频转码及视频处理能力，包括获取视频信息、视频拼接、视频水印、视频截图、视频剪辑、视频鉴黄、视频 DRM 加密等众多功能。支持多分辨率、多码率、多格式输出，率先支持 4K 分辨率输入/输出。视频云转码拥有一套完整的支持自定义的可配置化转码模板，支持快手在转码时对视频的参数进行自由设置。

通过腾讯通信云解决方案帮助实现快手业务需求，在 APP 内打开 H5 页面，提示用户发送特定格式的信息到指定号码，获取特定的信息；同时快手使用腾讯短信验证码服务进行用户身份验证，除简体中文以外，腾讯短信验证为快手支持繁体中文、英文、泰文、俄文、日文、韩文、印尼语、马来语、巴西葡萄牙语等不同的语言，满足快手全球化扩张的战略。

腾讯云为快手提供语音识别（语音转文本）、关键字检索与语音/图片鉴黄、人脸核身等智能 AI 服务。

（4）方案效果

通过腾讯云丰富的全球加速节点及海量云端存储服务，快手从容应对业务的高速扩展；通过植入腾讯云互动直播 SDK，快手快速上线高性能互动直播业务，为终端用户提供差异化服务；同时腾讯云强大的转码能力、完善的云通信解决方案及先进的云智能 AI 服务，都为快手的高速发展提供了助燃剂。

项目实训

实训题目

调研快手直播的业务范围、服务群体、功能特色及用户体验。

实训目的

锻炼调研能力、深入理解行业特点。

实训内容

撰写快手直播调研报告，报告涉及的内容：发行平台、推广、产品特点、业务范围、服务群体、用户体验、优缺点等。

任务 1.2　GoGal 小直播技术调研

微课 1-2
GoGal 小直播技术调研

任务描述

调研腾讯云 GoGal 小直播的基本原理、基础架构及业务流程。

知识学习

GoGal 够格直播电商系统，是一款基于多终端的直播电商交易平台。该平台提供创新式的“直播+电商”购物体验，以及全新的行业解决方案，以视频直播的形式向用户展示全球好物，并能直接下单购买。对于个人可以直播售卖自己商品。对于品牌商可创建商家账号，入驻本平台，并能挑选红人进行分销代售品牌商品，为商家解决流量变现、内容输出、品牌提升等问题。平台已获得腾讯、迅雷、京东、阿里等公司的直播技术及资源支持。

GoGal 够格直播的特点如下。

- 网红直播电商一体化商业模式。
- 功能丰富的直播电商 APP 软件。
- 强大便捷的运维后台系统。

任务实现

小赵所在的公司与腾讯是战略合作伙伴，在公司领导的帮助下，小赵联系上了负责腾讯云市场推广的某位前辈。在其协助下，小赵最终完成了调研，并整理成笔记，内容如下。

1. 视频直播原理及架构

(1) 基本原理

基本原理如图 1-2 所示。

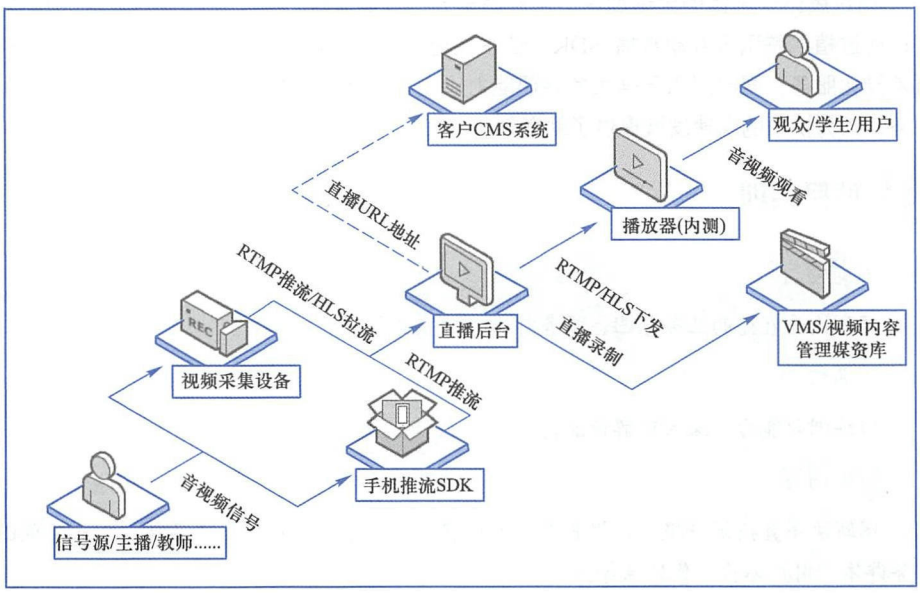


图 1-2
直播基本原理

视频直播基本原理分为以下两个阶段。

① 推流阶段：主播端开启直播，视频采集设备或手机推流 SDK 采集主播端信号源，再经 RTMP（推流协议）/HLS（拉流协议）传到直播后台。

② 下发阶段：直播后台经 RTMP/HLS 下发到观众端的播放器，形成音/视频。

（2）常用架构

常用架构如图 1-3 所示。

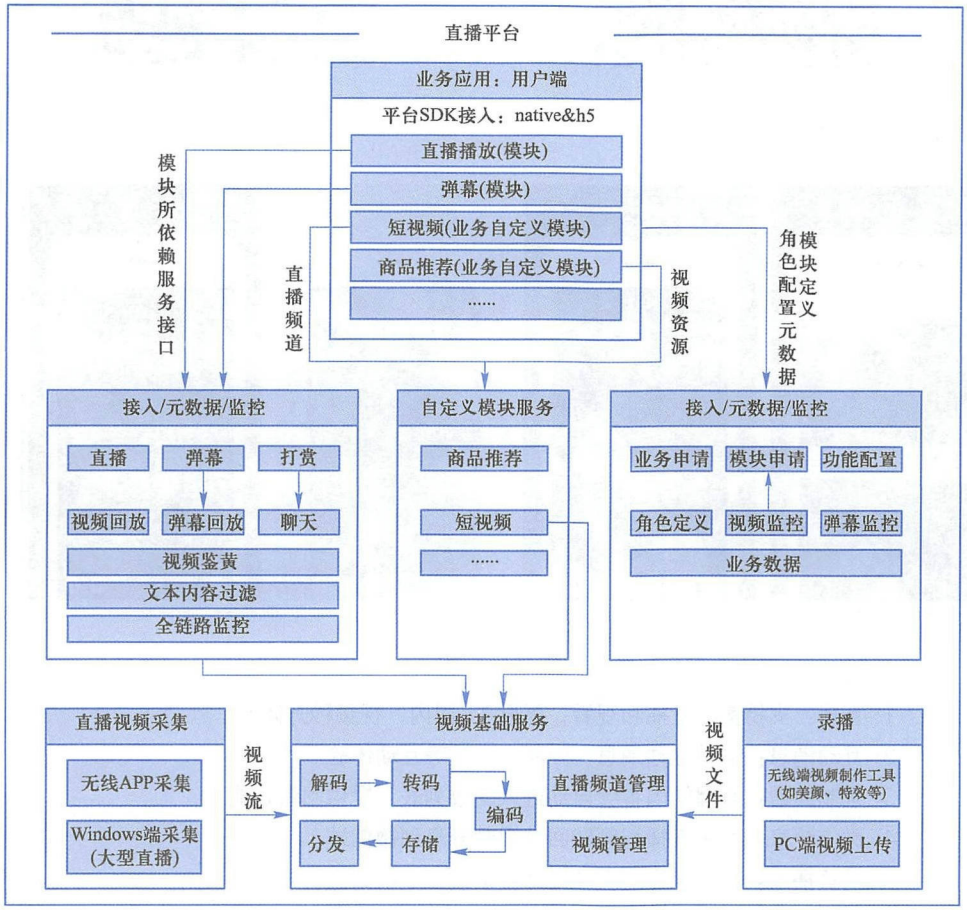


图 1-3
常用架构

可将视频直播分为业务应用层、SDK 服务层和视频基础服务层。其中：

① 业务应用层集中在用户端开发，对接平台 SDK 服务，实现直播相关的功能。

② SDK 服务层分为基础服务、接入/元数据/监控、自定义模块服务。主要提供基础 SDK 服务。

③ 视频基础服务层主要解决视频流的采集、解转编码、存储分发等视频文件相关的问题。

2. 腾讯云直播体验

（1）云直播 APP

云直播是基于腾讯视频云服务实现的一个直播 APP 的原型软件，开放终端和后台的实现源码，适合零基础的客户进行参考和评估之用。

扫描如图 1-4 和图 1-5 所示的二维码可下载“GoGal 小直播”体验版。

图 1-4
Android 下载

图 1-5
iOS 下载

界面截图如图 1-6 所示。

图 1-6
界面截图

支持以下功能。

① 直播：支持实时直播和观看，延迟 3 s 以内，视频秒开体验。

② 互动消息：支持互动消息、弹幕、点赞等互动体验。

③ 录制回看：直播的内容会自动进入回放列表，提供回看功能。

④ 微信分享：可将直播视频和回放视频分享到微信端。

⑤ 网页播放：可实现在网页端打开直播或回放视频进行观看。

⑥ 实时统计：可在直播间实时显示观看人数、点赞数等统计的数据。

（2）直播后台服务

云直播的后台服务目前部署于一台普通的腾讯云虚拟主机上，基于 Java 技术构建，提供源码下载和参考。

主要实现的服务如下。

① 直播列表：实现后台可实时查看所有直播和回放的视频列表。

② 实时统计：后台可实时统计视频观看人数、点赞数等信息。

③ 安全保护：使用腾讯云后台 API 接口的相关安全保护机制。

④ 运营监管：能够监控直播间运营相关的数据和视频。

⑤ 负载均衡：使用腾讯云的容灾及平行扩展机制。

项目实训

实训题目

下载并体验 GoGal 小直播。

实训目的

从用户角度出发，思考产品设计的特点。

实训内容

下载腾讯云 GoGal 小直播，体验视频直播，撰写体验心得，心得内容包括体验好坏、产品设计的优缺点等。

单元小结

本单元针对视频直播的基本概念、类型、原理及其框架产开了初步调研，同时调查了基于腾讯云直播的一些成功案例和一个腾讯云直播的 Demo。本单元旨在锻炼读者的调研能力，并向读者普及视频直播的一些基本知识，为以后的开发奠定基础。

单元 2 功能设计和服务租赁

学习目标

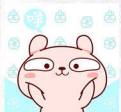
PPT2:
功能设计和服务租赁

知识目标

- 了解系统架构及 Web 工程目录。
- 了解 Bootstrap3 相关知识。
- 了解 Spring 相关知识。
- 了解 MyBatis 相关知识。
- 了解云直播所需服务的业务流程。

技能目标

- 了解云直播功能的简单设计思路。
- 掌握 Spring + Spring MVC + MyBatis + Bootstrap 整合框架的搭建方法。
- 掌握云直播所需服务的租赁流程。



学习情境

小赵的领导已经决定使用腾讯云的直播服务开发一款视频直播应用，取名为斗学直播 APP（下文简称斗学直播）。现在需要对整套系统进行设计，包括整体设计与技术选型，同时还需要租赁腾讯云的一些服务。公司将这些需求交给了公司的开发工程师小赵，通过调研小赵了解到 Spring + Spring MVC + MyBatis + Bootstrap 整合框架可以满足需求，于是小赵决定选用此框架来实现后台。

任务 2.1　功能设计和技术选型

任务描述

1. 参考 GoGal 小直播技术调研中的视频直播原理及架构的基础，对斗学直播的功能进行设计。

2. 调研 Bootstrap、Spring、MyBatis 的 3 个框架的由来及优势，掌握 3 个框架的使用方法，搭建 Spring + Spring MVC + MyBatis + Bootstrap 整合框架。

知识学习

1. Bootstrap

Bootstrap 是一个用于快速开发 Web 应用程序和网站的前端框架。在现代 Web 开发中，有几个几乎所有的 Web 项目中都需要的组件。Bootstrap 为开发者提供了所有这些基本的模块 Grid、Typography、Tables、Forms、Buttons 和 Responsiveness。此外，还有大量其他有用的前端组件，如 Dropdowns、Navigation、Modals、Typehead、Pagination、Carousal、Breadcrumb、Tab、Thumbnails、Headers 等。

有了这些组件，就可以搭建起来一个 Web 项目，并让它运行得更快速、更轻松。通过 Bootstrap 框架，开发者可以不需要知道太多的 HTML+CSS 知识就可以构建一个网站。但是，Bootstrap 是一个通用的框架，就像任何其他通用的东西，需要定制才能让它具有独特性。当开发者要定制时，就需要深入研究，没有良好的 HTML+CSS 基础是不可行的。

以下介绍项目当中会用到的网格系统布局、组件及 CSS 样式。在 Web 云存储中主要用到了 12 列布局、表格、按钮、列表。

注意

之所以首先说明结构，是为了简化开发过程，以及项目开发的编写规范。

（1）Bootstrap 的网格系统

Bootstrap 提供了一套响应式、移动设备优先的流式网格系统，随着屏幕的拉伸，系统会自动分为最多 12 列，如图 2-1 所示。

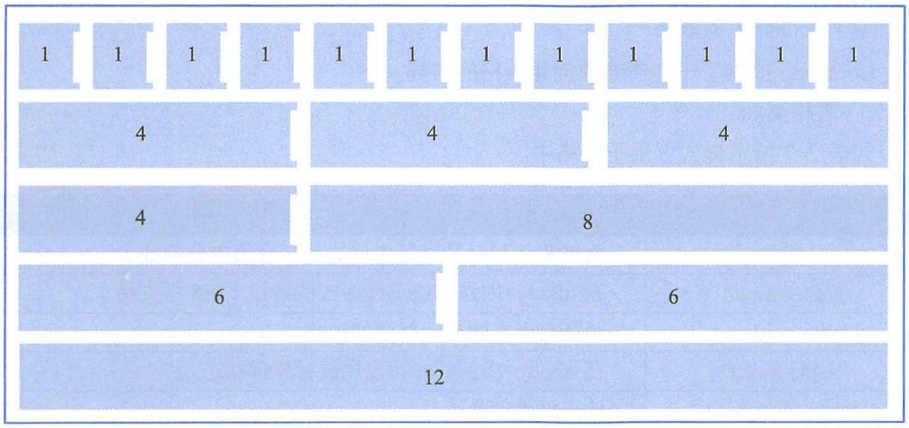


图 2-1
Bootstrap 的网格系统

① 网格选项。

如表 2-1 所示总结了 Bootstrap 网格系统如何跨多个设备工作。

网络选项	超小设备手机（<768 px）	小型设备平板电脑（≥768 px）	中型设备台式计算机（≥992 px）	大型设备台式计算机（≥1200 px）
网格行为	一直是水平的	以折叠开始，断点以上是水平的	以折叠开始，断点以上是水平的	以折叠开始，断点以上是水平的
最大容器宽度	None（auto）	750 px	970 px	1170 px
Class 前缀	.col-xs-	.col-sm-	.col-md-	.col-lg-
列数量和	12	12	12	12
最大列宽	Auto	60 px	78 px	95 px
间隙宽度	30 px（一个列的每边分别为 15 px）	30 px（一个列的每边分别为 15 px）	30 px（一个列的每边分别为 15 px）	30 px（一个列的每边分别为 15 px）
可嵌套	Yes	Yes	Yes	Yes
偏移量	Yes	Yes	Yes	Yes
列排序	Yes	Yes	Yes	Yes

表 2-1
网格系统

② 基本的网格结构。

下面是 Bootstrap 网格的基本结构。

```
<div class="container">
    <div class="row">
        <div class="col-*-*"></div>
        <div class="col-*-*"></div>
    </div>
    <div class="row">...</div>
</div>
```

（2）Bootstrap 的表格

Bootstrap 提供了一个清晰的创建表格的布局。

① 表格类。

如表 2-2 所示样式可用于表格中。

表 2-2
表格样式

类	描述
.table	为任意<table>添加基本样式（只有横向分隔线）
.table-striped	在<tbody>内添加斑马线形式的条纹（IE8 不支持）
.table-bordered	为所有表格的单元格添加边框
.table-hover	在<tbody>内的任一行启用鼠标悬停状态
.table-condensed	让表格更加紧凑

② <tr>、<th>和<td>类。

如表 2-3 所示的类可用于表格的行或者单元格。

表 2-3
提示样式

类	描述
.active	将悬停的颜色应用在行或者单元格上
.success	表示成功的操作
.info	表示信息变化的操作
.warning	表示一个警告的操作
.danger	表示一个危险的操作

③ 基本的表格。

如果只是需要一个只带有内边距（padding）和水平分割的基本表，可以添加 class .table，如下面实例所示。

```
<table class="table">
    <caption>基本的表格布局</caption>
<thead>
    <tr> <th>名称</th> <th>城市</th> </tr>
</thead>
<tbody>
    <tr> <td>Tanmay</td> <td>Bangalore</td> </tr>
    <tr> <td>Sachin</td> <td>Mumbai</td> </tr>
</tbody>
</table>
```

结果如图 2-2 所示。

图 2-2
效果图

基本的表格布局

名称	城市
Tanmay	Bangalore
Sachin	Mumbai

（3）Bootstrap 按钮

Bootstrap 提供了一些选项来定义按钮的样式，具体如表 2-4 所示。

以下样式可用于<a>、<button>或<input>元素上。

表 2-4 按钮类

类	描述
.btn	为按钮添加基本样式
.btn-default	默认/标准按钮
.btn-primary	原始按钮样式（未被操作）
.btn-success	表示成功的动作
.btn-info	该样式可用于要弹出信息的按钮
.btn-warning	表示需要谨慎操作的按钮
.btn-danger	表示一个危险动作的按钮操作
.btn-link	让按钮看起来像个超链接（仍然保留按钮行为）
.btn-lg	制作一个大按钮
.btn-sm	制作一个小按钮
.btn-xs	制作一个超小按钮
.btn-block	块级按钮（拉伸至父元素 100%的宽度）
.active	按钮被单击
.disabled	禁用按钮

下面的实例演示了上面所有的按钮 class。

```
<!-- 标准的按钮 -->
<button type="button" class="btn btn-default">默认按钮</button>
<!-- 提供额外的视觉效果，标识一组按钮中的原始动作 -->
<button type="button" class="btn btn-primary">原始按钮</button>
<!-- 表示一个成功的或积极的动作 -->
<button type="button" class="btn btn-success">成功按钮</button>
<!-- 信息警告消息的上下文按钮 -->
<button type="button" class="btn btn-info">信息按钮</button>
<!-- 表示应谨慎采取的动作 -->
<button type="button" class="btn btn-warning">警告按钮</button>
<!-- 表示一个危险的或潜在的负面动作 -->
<button type="button" class="btn btn-danger">危险按钮</button>
<!-- 并不强调是一个按钮，看起来像一个超链接，但同时保持按钮的行为 -->
<button type="button" class="btn btn-link">链接按钮</button>
```

结果如图 2-3 所示。

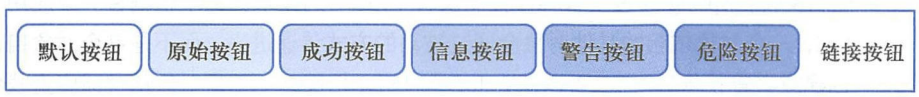


图 2-3 按钮样式

（4）Bootstrap 按钮组

按钮组允许多个按钮被堆叠在同一行上。当开发者需要把按钮对齐在一起时，这就显得非常有用。可以通过Bootstrap 按钮（Button）插件添加可选的 JavaScript 单选框和复选框样式行为。

下面的表 2-5 总结了 Bootstrap 提供的使用按钮组的一些重要的 class。

表 2-5
按钮组类

Class	描述
.btn-group	该 class 用于形成基本的按钮组。在.btn-group 中放置一系列带有 class.btn 的按钮
.btn-toolbar	该 class 有助于把几组 结合到一个 中，一般获得更复杂的组件
.btn-group-lg,.btn-group-sm, .btn-group-xs	这些 class 可应用到整个按钮组的大小调整，而不需要对每个按钮进行大小调整
.btn-group-vertical	该 class 让一组按钮垂直堆叠显示，而不是水平堆叠显示

（5）基本的按钮组

下面的实例演示了上面表格中讨论到的 class.btn-group 的使用。

```
<div class="btn-group">
    <button type="button" class="btn btn-default">按钮 1</button>
    <button type="button" class="btn btn-default">按钮 2</button>
    <button type="button" class="btn btn-default">按钮 3</button>
</div>
```

结果如图 2-4 所示。

图 2-4
按钮组

2. Spring

Spring 是一个开源框架，由 Rod Johnson 创建。它是为了解决企业应用开发的复杂性而创建的。Spring 使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。然而，Spring 的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何 Java 应用都可以从 Spring 中受益。

实际上，Spring 是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。Spring 框架的特点包括：

- 轻量：从大小与开销两方面而言，Spring 都是轻量的。完整的 Spring 框架可以在一个大小只有 1 MB 多的 JAR 文件里发布。并且 Spring 所需的处理开销也是微不足道的。此外，Spring 是非侵入式的。典型地，Spring 应用中的对象不依赖于 Spring 的特定类。
- 控制反转：Spring 通过一种称作控制反转（IoC）的技术促进了松耦合。当应用了 IoC，一个对象依赖的其他对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。可以认为 IoC 与 JNDI 相反——不是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。

- 面向切面：Spring 提供了面向切面编程的丰富支持，允许通过分离应用的业务逻辑与系统级服务（如审计和事务管理）进行内聚性的开发。应用对象只实现它们应该做的“完成业务逻辑”仅此而已。它们并不负责（甚至是意识）其他的系统级关注点，例如日志或事务支持。
- 容器：Spring 包含并管理应用对象的配置和生命周期，在这个意义上它是一种容器，开发者可以配置每个 Bean 如何被创建，以及它们是如何相互关联的。然而，Spring 不应该被混同于传统的重量级的 EJB 容器，它们经常是庞大与笨重的，难以使用。
- 框架：Spring 可以将简单的组件配置、组合成复杂的应用。在 Spring 中，应用对象被声明式地组合，是在一个典型的 XML 文件里。Spring 也提供了很多基础功能（事务管理、持久化框架集成等），将应用逻辑的开发留给了开发者。

所有 Spring 的这些特征使开发者能够编写更干净、更可管理并且更易于测试的代码。它们也为 Spring 中的各种模块提供了基础支持。

（1）Spring 包含的模块

Spring 框架由 7 个定义明确的模块组成，如图 2-5 所示。

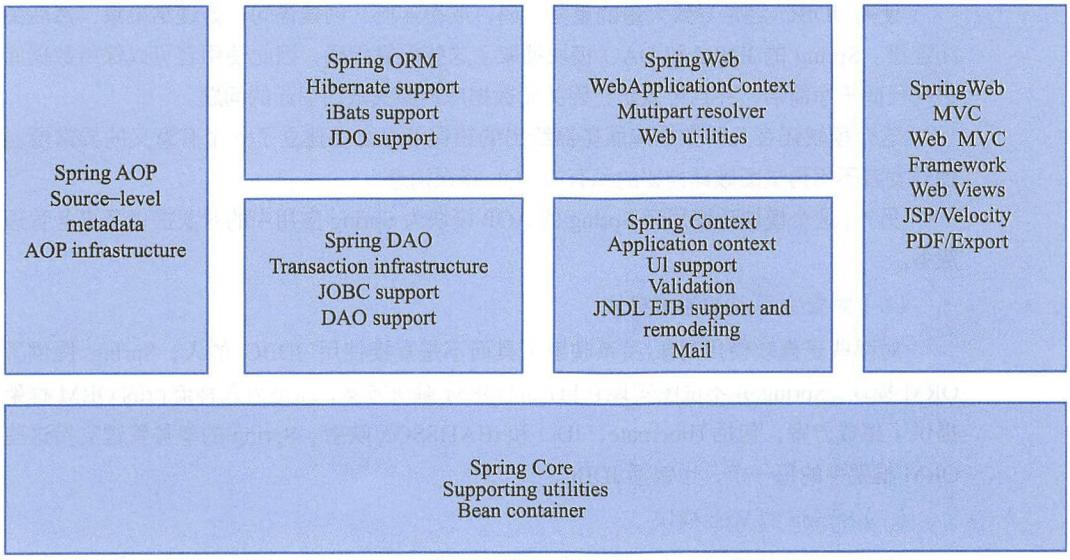


图 2-5 Spring 框架概览图

这些模块为开发者提供了开发企业应用所需的一切。但开发者不必将应用完全基于 Spring 框架，可以自由地挑选适合自己应用的模块而忽略其余的模块。所有的 Spring 模块都是在核心容器之上构建的。作为一名开发者，最可能对影响容器所提供服务的其他模块感兴趣。这些模块将会提供用于构建应用服务的框架，如 AOP 和持久性。

（2）核心容器

这是 Spring 框架最基础的部分，它提供了依赖注入（Dependency Injection）特征来实现容器对 Bean 的管理。这里最基本的概念是 BeanFactory，它是任何 Spring 应用的核心。BeanFactory 是工厂模式的一个实现，它使用 IoC 将应用配置和依赖说明从实际的应用代码中分离出来。

（3）应用上下文（Context）模块

核心模块的 BeanFactory 使 Spring 成为一个容器，而上下文模块使它成为一个框架。

笔 记

这个模块扩展了 BeanFactory 的概念，增加了对国际化（I18N）消息、事件传播及验证的支持。

另外，这个模块提供了许多企业服务，例如电子邮件、JNDI 访问、EJB 集成、远程及时序调度（Scheduling）服务，也包括对模版框架例如 Velocity 和 FreeMarker 集成的支持。

（4）Spring 的 AOP 模块

Spring 在它的 AOP 模块中提供了对面向切面编程的丰富支持。这个模块是在 Spring 应用中实现切面编程的基础。为了确保 Spring 与其他 AOP 框架的互用性，Spring 的 AOP 支持基于 AOP 联盟定义的 API。AOP 联盟是一个开源项目，它的目标是通过定义一组共同的接口和组件来促进 AOP 的使用，以及不同的 AOP 实现之间的互用性。可以通过访问相关站点，可以找到关于 AOP 联盟的更多内容。

Spring 的 AOP 模块也将元数据编程引入了 Spring。使用 Spring 的元数据支持，可以为开发者的源代码增加注释，指示 Spring 在何处及如何应用切面函数。

（5）JDBC 抽象和 DAO 模块

使用 JDBC 经常导致大量的重复代码，取得连接、创建语句、处理结果集，然后关闭连接。Spring 的 JDBC 和 DAO 模块抽取了这些重复代码，因此使用者可以保持数据库访问代码干净简洁，并且可以防止因关闭数据库资源失败而引起的问题。

这个模块还在几种数据库服务器给出的错误消息之上建立了一个有意义的异常层。使开发者不用再试图破译神秘的私有的 SQL 错误消息。

另外，这个模块还使用了 Spring 的 AOP 模块为 Spring 应用中的对象提供了事务管理服务。

（6）对象/关系映射集成模块

对那些更喜欢使用对象/关系映射工具而不是直接使用 JDBC 的人，Spring 提供了 ORM 模块。Spring 并不试图实现它自己的 ORM 解决方案，而是为几种流行的 ORM 框架提供了集成方案，包括 Hibernate、JDO 和 iBATISSQL 映射。Spring 的事务管理支持这些 ORM 框架中的每一个，也包括 JDBC。

（7）Spring 的 Web 模块

Web 上下文模块建立于上下文模块之上，提供了一个适合 Web 应用的上下文。另外，这个模块还提供了一些面向服务的支持。例如，实现文件上传的 multipart 请求，它也提供了 Spring 和其他 Web 框架的集成，如 Struts、WebWork。

（8）Spring 的 MVC 框架

Spring 为构建 Web 应用提供了一个功能全面的 MVC 框架。虽然 Spring 可以很容易地与其他 MVC 框架集成，如 Struts。但 Spring 的 MVC 框架使用 IoC 对控制逻辑和业务对象提供了完全的分离。

它也允许用户声明性地将请求参数绑定到用户的业务对象中。此外，Spring 的 MVC 框架还可以利用 Spring 的任何其他服务，如国际化信息与验证。

Spring 框架内容涉及比较多，这里不再赘述，仅介绍项目中涉及的相关技术。

（9）Spring Web MVC 架构

Spring Web MVC 框架也是一个基于请求驱动的 Web 框架，并且也使用了前端控制器

模式来进行设计，再根据请求映射规则分发给相应的页面控制器（动作/处理器）进行处理。首先整体看一下 Spring Web MVC 处理请求的流程。

如图 2-6 所示的是 MVC 处理请求的流程图。

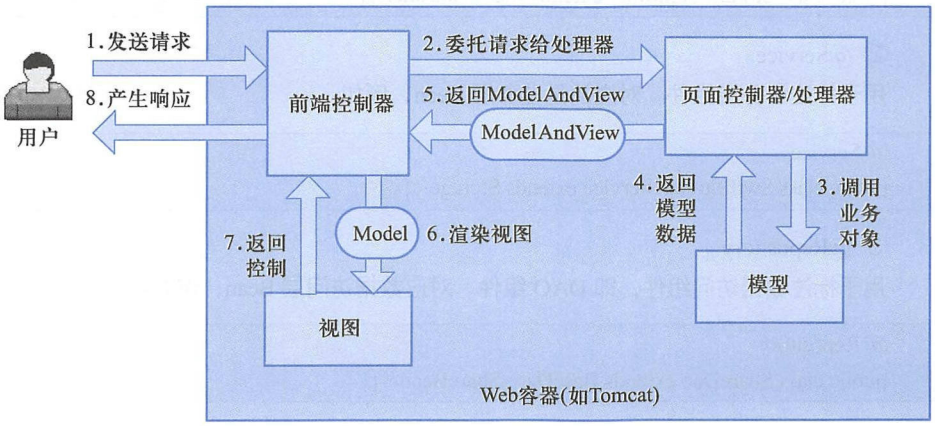


图 2-6
MVC 处理请求的流程图

具体执行步骤如下。

① 首先用户发送请求至前端控制器，前端控制器根据请求信息（如 URL）来决定选择哪一个页面控制器进行处理并把请求委托给它，即图 2-7 中的步骤 1 和步骤 2。

② 页面控制器接收到请求后，进行功能处理，首先需要收集和绑定请求参数到一个对象（这个对象在 Spring Web MVC 中被称为命令对象），并进行验证，然后将命令对象委托给业务对象进行处理。处理完毕后返回一个 ModelAndView（模型数据和逻辑视图名），即图 2-7 中的步骤 3、步骤 4 和步骤 5。

③ 前端控制器收回控制权，然后根据返回的逻辑视图名，选择相应的视图进行渲染，并把模型数据传入以便视图渲染，即图 2-7 中的步骤 6 和步骤 7。

④ 前端控制器再次收回控制权，将响应返回给用户，即图 2-7 中的步骤 8。至此整个执行过程结束。

ModelAndView 代码示例如下。

```
public ModelAndView getModelAndView() {
        ModelAndView view = new ModelAndView();
view.addObject("path","/");
        view.setViewName("/main");
        return view;
}
```

（10）Spring 注解

目前注解非常流行，很多主流框架都支持注解，编写代码的时候尽量使用注解，一是方便，二是代码更加简洁。Spring 框架里面也有很多注解，下面主要介绍项目中涉及的几类注解。

① @Controller。

用于标注控制层组件。在 SpringMVC 中，控制器 Controller 负责处理由 DispatcherServlet 分发的请求，它把用户请求的数据经过业务处理层处理之后封装成一个 Model，然后再把该 Model 返回给对应的 View 进行展示。对应表现层的 Bean，也就是

Action，例如：

```
@Controller
public class StorageController extends BaseController {}
```

② @Service。

用于标注业务层组件。对应的是业务层 Bean，例如：

```
@ Service
public class SwiftStorageService extends Storage {}
```

③ @Repository。

用于标注数据访问组件，即 DAO 组件。对应数据访问层 Bean，例如：

```
@ Repository
public class ShareDao extends BaseDao<ShareBean> {}
```

④ @Autowired。

自动装配，它可以对类成员变量、方法及构造函数进行标注，完成自动装配的工作。例如：

```
@Autowired
private ShareService shareService;
```

⑤ @RequestMapping。

一个用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中所有响应请求的方法都是以该地址作为父路径。例如：

```
@RequestMapping("/home")
public ModelAndView home(HttpServletRequest request, HttpServletResponse response, String path){
}
```

说明

参数绑定说明：如果使用以下 URL 请求：http://localhost/home，将会对应访问到 home 方法。

⑥ @ResponseBody。

它的作用是将返回类型直接输入到 HTTP response body 中。

在输出 JSON 格式的数据时会用到。例如：

```
@RequestMapping("/login")
@ResponseBody
public Object login(HttpServletRequest request, HttpServletResponse response, String username, String password){
    return new MessageBean(true,Constants.SUCCESS_1);
}
```

在用 Ajax 异步提交数据的时候，后台对接的方法需要加上@ResponseBody 注解。

3. MyBatis

MyBatis 是支持定制化 SQL、存储过程及高级映射的优秀的持久层框架。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数及获取结果集。MyBatis 可以对配置和原生 Map 使用简单的 XML 或注解将接口和 Java 的 POJOs（Plain Old Java Objects，普通的 Java 对象）映射成数据库中的记录。MyBatis 具有以下两个优点：

（1）实现了接口绑定，使用更加方便

在 ibatis2.x 中需要在 DAO 的实现类中指定具体对应哪个 XML 映射文件，而 MyBatis 实现了 DAO 接口与 XML 映射文件的绑定，自动生成接口的具体实现，使用起来变得更加省事和方便。

（2）对对象关系映射进行改进，效率更高

MyBatis 采用功能强大的基于 OGNL 的表达式来消除其他元素。

MyBatis 的框架架构如图 2-7 所示。

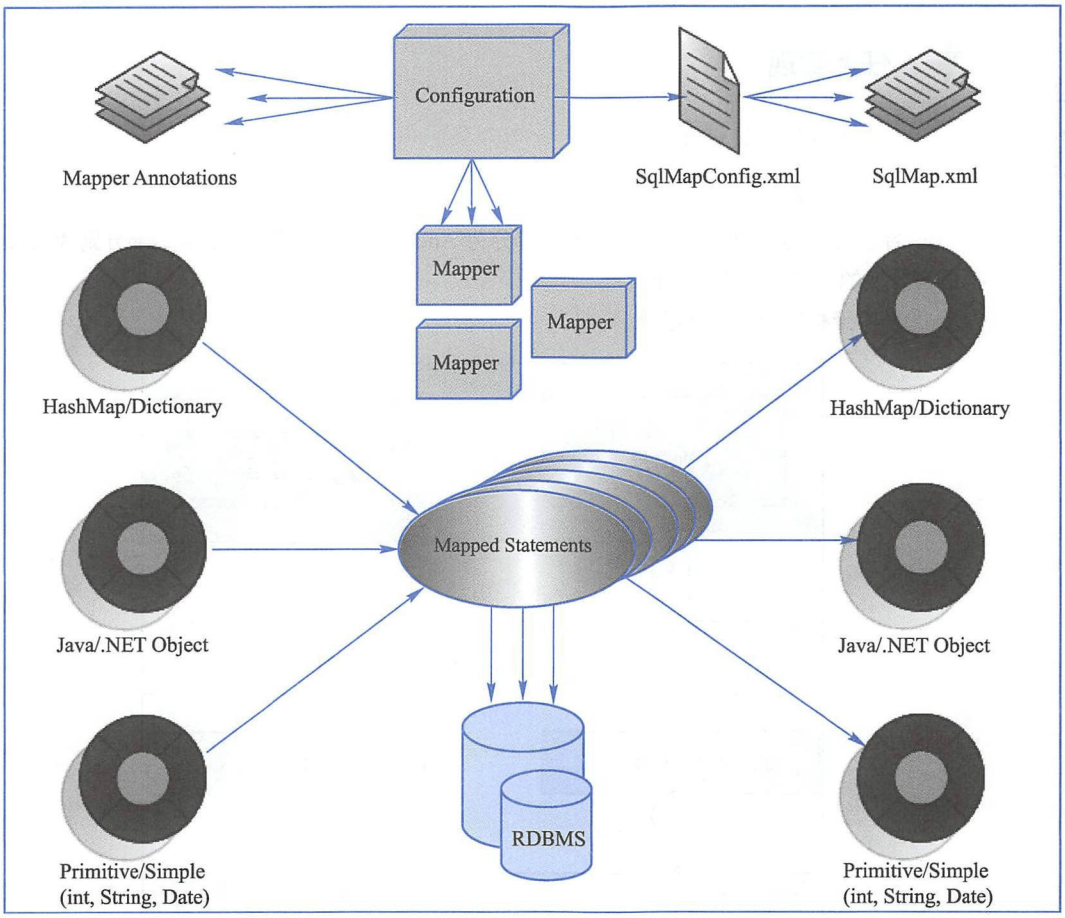


图 2-7
MyBatis 架构

看到 MyBatis 的框架图，可以清晰地看到 MyBatis 的整体核心对象，如图 2-8 所示。

非卖品，仅供非商业用途或交流学习使用

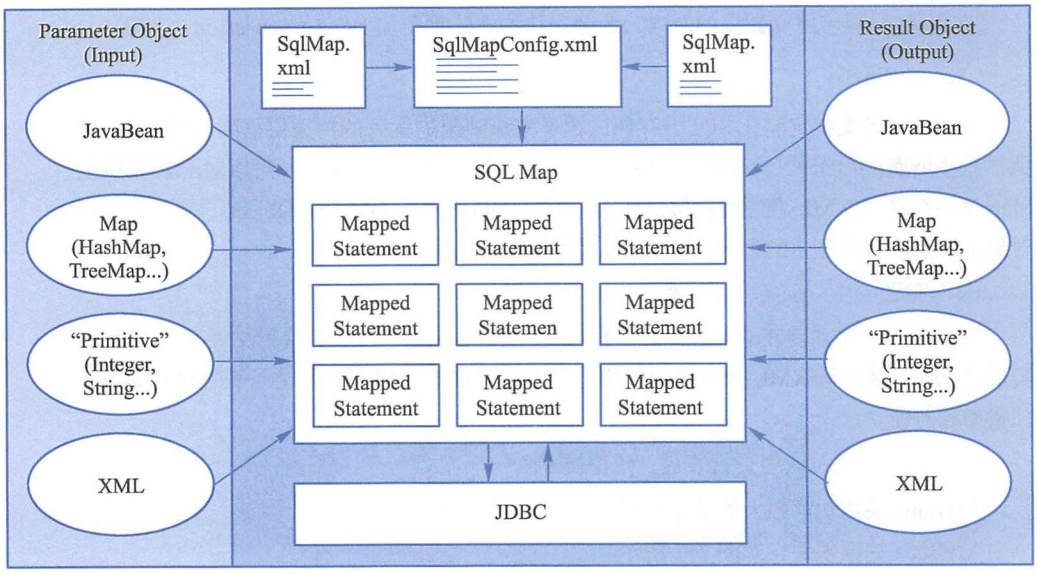


图 2-8
MyBatis 对象管理

任务实现

微课 2-1
功能设计和技术选型

1. 系统架构设计和功能设计

(1) 斗学直播的系统架构设计

斗学直播的系统分为两部分，第一部分为 Android 平台的客户端，第二部分则为 Java 语言开发的直播后台管理系统。

系统架构图如图 2-9 所示。

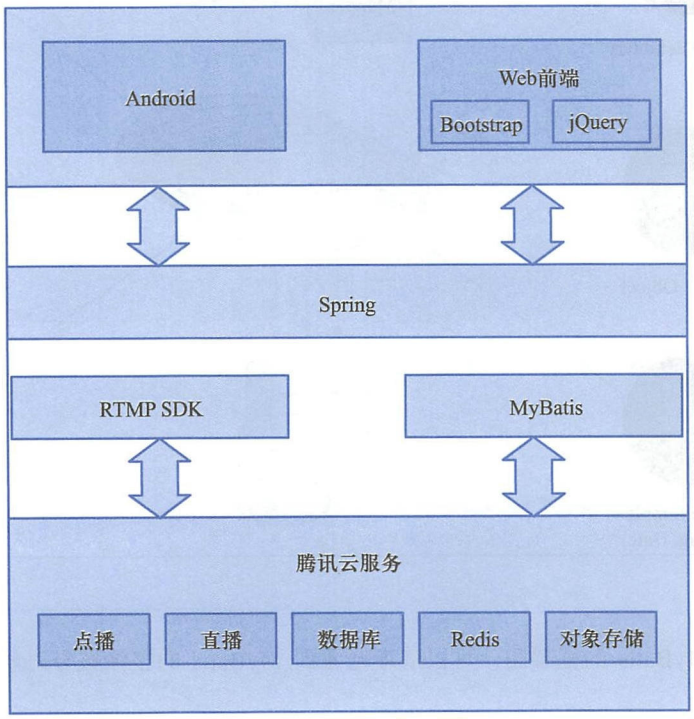


图 2-9
系统架构图

非卖品，仅供非商业用途或交流学习使用

Web 工程的目录结构如图 2-10 所示。

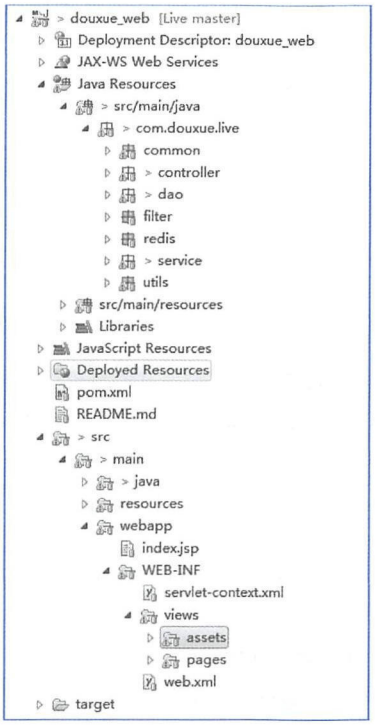


图 2-10
Web 工程的目录结构

（2）斗学直播的功能设计

具体功能任务如表 2-6 所示。

表 2-6
小直播功能列表

模块	操作	说明
用户管理模块	用户注册	用于用户注册
	用户登录	用于用户登录
	个人中心	用于修改个人基本信息
直播模块	开启直播	用于用户在手机中开启直播
	直播列表	用于手机 APP 中显示直播列表
	观看直播	用于手机用户观看直播
直播互动模块	点赞	用于用户在直播间点赞
	弹幕	用于在直播间发送文字
	功能扩展	闪光灯、美颜、摄像头翻转
直播后台管理	登录注销	管理员用户的登入与注销
	直播管理	直播间列表、禁播、监控、搜索
	用户管理	用户管理列表、禁用、启用、删除

2. 构建 Spring + Spring MVC + MyBatis + Bootstrap 整合框架

在了解了框架的基础知识之后，小赵开始自己动手搭建 Spring + Spring MVC +

非卖品，仅供非商业用途或交流学习使用

源码 2-1：
功能设计和技术选型

MyBatis + Bootstrap 整合框架。

① 打开 Eclipse 导入本教材提供的配套项目包，此项目包是一个没有实现具体功能的空工程。

在/douxue_web/src/main/webapp/WEB-INF 文件夹下的 web.xml 补充相关控制器，具体代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    id="WebApp_ID" version="3.1">
    <display-name>douxue_web</display-name>
    <filter>
      <filter-name>CharsetEncodingFilter</filter-name>

      <filter-class>com.douxue.live.common.CharsetEncodingFilter</filter-class>
      <init-param>
        <param-name>encoding</param-name>
        <param-value>utf-8</param-value>
      </init-param>
    </filter>
    <filter-mapping>
      <filter-name>CharsetEncodingFilter</filter-name>
      <url-pattern>/*</url-pattern>
    </filter-mapping>

    <filter>
        <filter-name>AuthFilter</filter-name>
        <filter-class>com.douxue.live.filter.AuthFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>AuthFilter</filter-name>
        <servlet-name>dispatcher</servlet-name>
    </filter-mapping>

    <context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath*:applicationContext.xml</param-value>
     </context-param>

  <listener>
```

非卖品，仅供非商业用途或交流学习使用


```
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>

  <!-- Web 前端控制器  -->
  <servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/servlet-context.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

  <!-- 配置 session 超时时间，单位：分钟 -->
     <session-config>
          <session-timeout>30</session-timeout>
     </session-config>

    <welcome-file-list>
          <welcome-file>index.jsp</welcome-file>
     </welcome-file-list>
</web-app>
```

② 在 src/main/webapp/WEB-INF 目录下新建 servlet-context.xml 文件，过滤资源文件，具体代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-4.1.xsd
```

非卖品，仅供非商业用途或交流学习使用


```
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-4.1.xsd">
<!-- Scans the classpath of this application for @Components to deploy as beans -->

<context:component-scan base-package="com.douxue.live.controller" />
<!--  -->
<mvc:annotation-driven  validator="validator"/>

<!-- 用户登录验证拦截器 -->
<mvc:interceptors>
    <mvc:interceptor>
        <mvc:mapping path="/**" />
        <mvc:exclude-mapping path="/assets/**" />
        <!-- <mvc:exclude-mapping path="assets/**" /> -->
        <mvc:exclude-mapping path="/app/**" />
        <mvc:exclude-mapping path="/web/admin/login" />
        <mvc:exclude-mapping path="/web/admin/loginPage" />
        <bean class="com.douxue.live.filter.WebLoginFilter" />
    </mvc:interceptor>
</mvc:interceptors>

<!-- Configures the @Controller programming model -->
<!-- Resolves view names to protected .jsp resources within the /WEB-INF/views directory -->
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/views/" />
    <property name="suffix" value=".jsp" />
</bean>
<!-- 静态资源映射 -->
<mvc:resources mapping="/assets/**" location="/WEB-INF/views/assets/"/>
<!-- <mvc:resources mapping="assets/**" location="/WEB-INF/views/assets/"/> -->

<bean
class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandler Mapping" />
<bean
class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter" >
    <property name="messageConverters">
        <list>
            <ref bean="mappingJacksonHttpMessageConverter"/>
        </list>
    </property>
</bean>
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用


```
    <bean      id="mappingJacksonHttpMessageConverter"
    class="org.springframework.http. converter.json.MappingJackson2HttpMessageConverter">
        <property name="supportedMediaTypes">
            <list>
                <value>text/html;charset=UTF-8</value>
            </list>
        </property>
    </bean>
    <bean
    id="validator" class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean">
        <!-- 指定校验器由谁提供 -->
        <property name="providerClass" value="org.hibernate.validator.HibernateValidator">
</property>
        <property name="validationMessageSource" ref="validationMessageSource"></property>
    </bean>
    <bean id="validationMessageSource" class="org.springframework.context.support.Reloadable
ResourceBundleMessageSource">
        <!-- 指定校验文件的地址，该文件默认就是 properties 文件 -->
        <property name="basenames">
            <list>
                <value>classpath:validationMessage</value>
            </list>
        </property>
        <property name="fileEncodings" value="utf8"></property>
    </bean>
    </beans>
```

③ 在 src/main/resources 目录下新建 applicationContext.xml，设置应用配置，具体代码如下。

```
    <?xml version="1.0" encoding="utf-8"?>
    <beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:p="http://www.springframework.org/schema/p"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xmlns:task="http://www.springframework.org/schema/task"
        xmlns:tx="http://www.springframework.org/schema/tx"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.0.xsd
```


版权所有，严禁传播，违者自负法律责任！

版权所有，严禁传播，违者自负法律责任！


```
                http://www.springframework.org/schema/data/mongo
                http://www.springframework.org/schema/data/mongo/spring-mongo-1.0.xsd
                http://www.springframework.org/schema/tx
                http://www.springframework.org/schema/tx/spring-tx.xsd
                http://www.springframework.org/schema/task
                http://www.springframework.org/schema/task/spring-task-3.0.xsd
                http://www.springframework.org/schema/mvc
                http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd
                    ">

                <bean id="propertyConfigurer"
                        class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
                        <property name="order" value="10" />
                        <property name="ignoreUnresolvablePlaceholders" value="true" />
                        <property name="locations">
                                <list>
                                        <value>classpath:application.properties</value>
                                </list>
                        </property>
                        <property name="fileEncoding" value="utf-8" />
                </bean>

                <context:component-scan base-package="com.douxue.live.dao.mapper" />
                <context:component-scan base-package="com.douxue.live.service" />
                <context:component-scan base-package="com.douxue.live.dao.redis" />
                <context:component-scan base-package="com.douxue.live.controller" />

                <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
                        <property name="basePackage" value="com.douxue.live.dao.mapper" />
                        <property name="sqlSessionFactoryBeanName"
value="myBatisSqlSession Factory" />
                        <property name="annotationClass"
value="org.springframework.stereotype. Repository" />
                </bean>

                <bean id="myBatisSqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactory Bean">
                        <property name="dataSource" ref="dataSource" />
                        <property name="configLocation" value="classpath:mybatis-config.xml" />
                        <property name="mapperLocations" value="classpath*:mapper/*Mapper.xml" />
                </bean>
```


非卖品，仅供非商业用途或交流学习使用

非卖品，仅供非商业用途或交流学习使用


```
        <!-- MyBatis 扫描路径 -->
        <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
            <property name="basePackage" value="com.douxue.live.dao.mapper" />
            <property      name="sqlSessionFactoryBeanName"
value="myBatisSqlSession Factory" />
        </bean>

        <tx:annotation-driven />
        <bean id="transactionManager"
            class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
            <property name="dataSource" ref="dataSource" />
        </bean>

        <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"
            init-method="init" destroy-method="close">
            <!-- 基本属性 url、user、password -->
            <property name="url" value="${jdbc_url}" />
            <property name="username" value="${jdbc_username}" />
            <property name="password" value="${jdbc_password}" />

            <!-- 配置初始化大小、最小、最大 -->
            <property name="initialSize" value="1" />
            <property name="minIdle" value="1" />
            <property name="maxActive" value="20" />

            <!-- 配置获取链接等待超时的时间 -->
            <property name="maxWait" value="60000" />

            <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲链接，单位：毫秒 -->
            <property name="timeBetweenEvictionRunsMillis" value="60000" />

            <!-- 配置一个链接在池中最小生存的时间，单位：毫秒 -->
            <property name="minEvictableIdleTimeMillis" value="300000" />

            <property name="validationQuery" value="SELECT 'x'" />
            <property name="testWhileIdle" value="true" />
            <property name="testOnBorrow" value="false" />
            <property name="testOnReturn" value="false" />

            <!-- 打开 PSCache，并且指定每个链接上 PSCache 的大小 -->
            <property name="poolPreparedStatements" value="true" />
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用


```
                <property name="maxPoolPreparedStatementPerConnectionSize" value="20" />

                <!-- 配置监控统计拦截的 filters -->
                <property name="filters" value="stat" />
            </bean>

            <!-- 打开 SpringMVC 文件上传功能 -->
            <bean id="multipartResolver"
                class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
                <property name="defaultEncoding" value="UTF-8"/>
                <property name="maxUploadSize" value="2097153"/>
                <property name="maxInMemorySize" value="10240" />
            </bean>
            <import resource="classpath:spring-jedis.xml" />
        </beans>
```

④ 在 src/main/resources 目录下新建 generatorConfig.xml，配置生产者信息，具体代码如下。

```
        <?xml version="1.0" encoding="UTF-8"?>
        <!DOCTYPE generatorConfiguration
                PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
                "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">
        <generatorConfiguration>
            <!-- 数据库驱动 -->
            <classPathEntry location="C:\mysql-connector-java-5.1.38.jar" />
            <context id="DB2Tables" targetRuntime="MyBatis3">
                <commentGenerator>
                    <property name="suppressDate" value="true" />
                    <!-- 是否去除自动生成的注释，true：是；false：否 -->
                    <property name="suppressAllComments" value="true" />
                </commentGenerator>
                <!--数据库链接 URL，用户名、密码 -->
                <jdbcConnection driverClass="com.mysql.jdbc.Driver"
                    connectionURL="jdbc:mysql://57fc4fa14a8cb.gz.cdb.myqcloud.com:17068/
mz_live" userId="cdb_outerroot"
                    password="livedemo@mysql">
                </jdbcConnection>
                <javaTypeResolver>
                    <property name="forceBigDecimals" value="false" />
                </javaTypeResolver>
                <!-- 生成模型的包名和位置 -->
```


版权所有，严禁传播，违者自负法律责任！

```
            <javaModelGenerator targetPackage="com.douxue.live.dao.entity"
                targetProject="src/main/java">
                <property name="enableSubPackages" value="true" />
                <property name="trimStrings" value="true" />
            </javaModelGenerator>
            <!-- 生成映射文件的包名和位置 -->
            <sqlMapGenerator targetPackage="mapper" targetProject="src/main/resources">
                <property name="enableSubPackages" value="true" />
            </sqlMapGenerator>
            <!-- 生成 DAO 的包名和位置 -->
            <javaClientGenerator type="XMLMAPPER"
                targetPackage="com.douxue.live.dao.mapper"
targetProject="src/main/java">
                <property name="enableSubPackages" value="true" />
            </javaClientGenerator>
            <!-- 要生成哪些表 -->
            <!-- mybatis-generator:generate -->

            <table tableName="t_admin_user" domainObjectName="AdminUser"
                enableCountByExample="false" enableUpdateByExample="false"
                enableDeleteByExample="false" enableSelectByExample="false"
                selectByExampleQueryId="false">
            </table>
        </context>
    </generatorConfiguration>
```

⑤ 在 src/main/resources 目录下新建 mybatis-config.xml，配置 MyBatis 信息，具体代码如下。

```
    <?xml version="1.0" encoding="UTF-8" ?>
    <!DOCTYPE configuration
        PUBLIC "-//ibatis.apache.org//DTD Config 3.0//EN"
        "http://ibatis.apache.org/dtd/ibatis-3-config.dtd">
    <configuration>
        <settings>
            <!-- 全局映射器启用缓存 -->
            <setting name="cacheEnabled" value="true" />
            <!-- 查询时，关闭关联对象即时加载以提高性能 -->
            <setting name="lazyLoadingEnabled" value="true" />
            <!-- 设置关联对象加载的形态，此处为按需加载字段（加载字段由 SQL 指
定），不会加载关联表的所有字段，以提高性能 -->
            <setting name="aggressiveLazyLoading" value="false" />
```


版权所有，严禁传播，违者自负法律责任！

```
        <!-- 对于未知的 SQL 查询，允许返回不同的结果集以达到通用的效果 -->
        <setting name="multipleResultSetsEnabled" value="true" />
        <!-- 允许使用列标签代替列名 -->
        <setting name="useColumnLabel" value="true" />
        <!-- 允许使用自定义的主键值（比如由程序生成的 UUID 32 位编码作为键值），数据表的 PK 生成策略将被覆盖 -->
        <setting name="useGeneratedKeys" value="true" />
        <!-- 给予被嵌套的 resultMap 以字段—属性的映射支持 -->
        <setting name="autoMappingBehavior" value="FULL" />
        <!-- 对于批量更新操作缓存 SQL 以提高性能，默认 SIMPLE 若设置为 BATCH，selectKEy 无法返回值 -->
        <setting name="defaultExecutorType" value="SIMPLE" />
        <!-- 数据库超过 25 000 秒仍未响应则超时 -->
        <setting name="defaultStatementTimeout" value="25000" />
    </settings>
    <plugins>
        <plugin interceptor="com.douxue.live.common.page.PageInterceptor">
            <property name="dbType" value="mysql" />
        </plugin>
    </plugins>
</configuration>
```

⑥ 在 src/main/resources/development 目录下新建 application.properties，配置系统参数信息，具体代码如下。

```
# Mysql
jdbc_url=jdbc:mysql://host:prot/db_name
jdbc_username=account
jdbc_password=password
# Redis
redis_uri=host
redis_port=prot
redis_password=password
redis_timeout=2000
redis_pool_maxIdle=100
redis_pool_maxTotal=1000
# 直播相关配置
# APP_ID
live_appid=xxxxxx
# BIZID
live_bizid=xxxx
# 推流防盗链 KEY
```


版权所有，严禁传播，违者自负法律责任！

```
live_push_safe_key=xxxxxxxxxxxxxxxxxxxxxxxxxxxx
# API 鉴权 KEY
live_api_sign_key=xxxxxxxxxxxxxxxxxxxxxxxxxxxx
# 直播回放切片生成格式
live_record=mp4
# 直播相关推/拉流地址
live_push_rtmp_url=rtmp://%s.livepush.myqcloud.com/live/%s?bizid=%s
live_play_rtmp_url=rtmp://%s.liveplay.myqcloud.com/live/%s
live_play_flv_url=http://%s.liveplay.myqcloud.com/live/%s.flv
live_play_hls_url=http://%s.liveplay.myqcloud.com/live/%s.m3u8
# 直播公共相关 API 地址
live_common_access_url=http://fcgi.video.qcloud.com/common_access
# 腾讯云存储
qcloud_cos_secret_id=xxxxxxxxxxxxxxxxxxxxxxxxxxxx
qcloud_cos_secret_key=xxxxxxxxxxxxxxxxxxxxxxxxxxxx
qcloud_cos_app_id=xxxxxxxxx
qcloud_cos_bucket=xxxxxxx
```

项目实训

实训题目

1. 画出斗学直播的功能模块图。
2. 探索 Web 框架。

实训目的

1. 熟悉功能模块图的规范画法，深刻理解功能模块划分的层级结构和设计思路。
2. 掌握 Bootstrap、Spring、MyBatis 的基础知识。
3. 掌握 Web 框架的搭建。

实训内容

1. 按照软件工程规范，根据斗学直播的功能设计表画出功能模块图，要求体现其分类要领和层级关系。
2. 构建 Web 开发框架 SSM（Sring+ Spring MVC + MyBatis）。

任务 2.2　服 务 租 赁

任务描述

调研云直播所需的服务及其购买流程，了解云直播各个服务的业务流程，购买云直播所需要的所有服务。

版权所有，严禁传播，违者自负法律责任！

知识学习

1. 直播服务

（1）基本描述

直播（Live Video Broadcasting）依托腾讯强大的技术平台，将腾讯视频等核心业务底层能力开放给用户，为用户提供专业、稳定、快速的直播接入和分发服务，全面满足超低延迟和超大并发访问量的苛刻要求，并提供腾讯自研的推流器和播放器 SDK，方便客户端定制自己的推流器和播放器 APP。它具有低延迟、高安全、高性能、易接入、多终端、多码率支持等特点。

（2）业务介绍

① 可靠转换。

提供快速可靠的格式转换；支持 FLV、RTMP 及 HLS 的下发，超低延迟。

② 超高容量支持。

可支持百万并发直播收看，海量 CDN 节点提供全方位覆盖。

③ 移动端推流器和播放器 SDK。

自研 Android/iOS 推流器和播放器 SDK，支持 RTMP 推流，支持 FLV、RTMP 直播，支持 FLV 点播。可方便实现移动端直播推流和播放功能，自有移动应用集成。

④ 频道管理 API。

提供直播频道创建、修改、查询等 API 功能，可无缝同自有视频内容管理系统整合集成。

⑤ 录制截图和多码率。

支持通过 API 对直播过程进行录制和截图；录制视频直接对接点播平台，截图图片直接存放面向对象存储的 COS（Cloud Object Service）系统，以备后续处理，高效便捷；支持多码率和水印功能。

⑥ 直播码管理方式。

支持直播码管理方式对大规模直播进行管理。

（3）产品构架

典型的产品构架如图 2-11 所示。

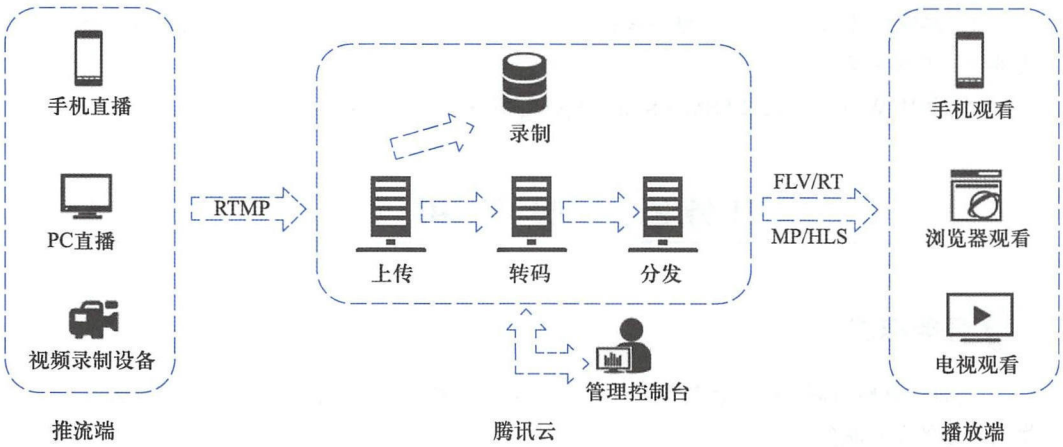


图 2-11
直播架构

版权所有，严禁传播，违者自负法律责任！

- 推流端：支持任意 RTMP 推流设备，包括使用集成腾讯自研直播 SDK 的 APP 或者第三方直播软硬件推流至直播系统。
- 播放端：可以使用集成腾讯自研直播 SDK 的 APP 或者第三方播放器。
- 管理组件：用户可通过 API、管理控制台方式，对直播活动进行管理，并查询相关统计数据。
- 分发组件：通过遍布全国的 CDN 系统，为直播用户提供快速灵活的播放方式。用户可使用自研或腾讯云提供的直播播放器 SDK 方便同自有业务集成，也可直接通过 H5 页面（HTML 5）收看 HLS 直播。

（4）主要功能

直播产品支持功能如表 2-7 所示。

表 2-7 直播功能

类别	功能名称	详细内容
直播接入	推流协议	支持 RTMP 推流
	推流软件	支持集成腾讯直播 SDK 的 APP 或常见第三方推流软件，包括 OBS、XSplit、FMLE 等
	推流设备	支持常见第三方 RTMP 推流硬件和编码盒
直播下发	下发协议	支持 RTMP、FLV 和 HLS 拉流
	下发原始码率规格	音/视频编码、码率和输入一致
	实时转码	支持实时转码：高清——分辨率 1280×720（码率 900 kb/s）；标清——分辨率 960×540（码率 550 kb/s）
直播收看	收看方软件	支持常见第三方 FLV、RTMP、HLS 播放器
	收看方 SDK	提供 Android、iOS 播放器 SDK（支持 FLV、RTMP）；提供 Web 播放器 SDK（支持 FLV、RTMP、HLS）
频道管理	管理方式	支持管理控制台图形化管理、API 管理；支持（域名+直播码）管理方式
	API 管理	创建、修改、删除、启用、停止直播频道；创建停止录制、创建停止截图任务等
	管理控制台	创建、修改、删除、启用、停止直播频道；统计查询；直播测试
	水印管理	支持在管理控制台或者 Android、iOS SDK 自定义水印样式和位置，推荐使用 SDK 设置水印
直播安全	黑白名单	支持接收方黑白名单 URL 管理自定义
	防盗链	支持推流、拉流 URL 防盗链
直播录制	API 录制	支持通过 API 录制直播过程并存储于点播平台
直播截图	API 截图	支持通过 API 对直播过程截图并存储于 COS 面向对象服务平台
统计	管理控制台统计	实时下行带宽、下行流量、直播在线数、直播请求数（运营商/地域维度）、直播频道统计
	API 统计	查询当前并发送收看数

（5）应用场景

直播软件可应用在各种丰富的场景，并可搭配腾讯云视频解决方案的其他产品，实现游戏直播、视频门户、社交直播等各种场景。

版权所有，严禁传播，违者自负法律责任！

1）游戏直播场景

主要特性如下。

- 满足标准大并发、低延迟直播要求。
- 借助超过 500+节点的直播发布平台，帮助游戏视频门户用户具备和腾讯视频底层服务水平一致的发布能力，全面快速地覆盖各地用户。
- 系统可轻松接入专业视频录制设备、播控平台等任意标准 HLS/RTMP 直播源，无论是人气游戏主播，还是竞技赛事，均可快速发布。
- 提供灵活的播放器，帮助快速通过移动应用、网页等各种展示方式全方位覆盖玩家。
- 详细的统计分析更可帮助游戏平台精准地把握玩家行为特征。
- 可实时录制并对接点播平台，提供完善视频录制回看服务。
- IM SDK 功能可集成在移动端和网页应用中，提供基于文本、图片的即时通信功能，丰富直播交互场景。

系统构架如下。

游戏直播系统架构如图 2-12 所示。

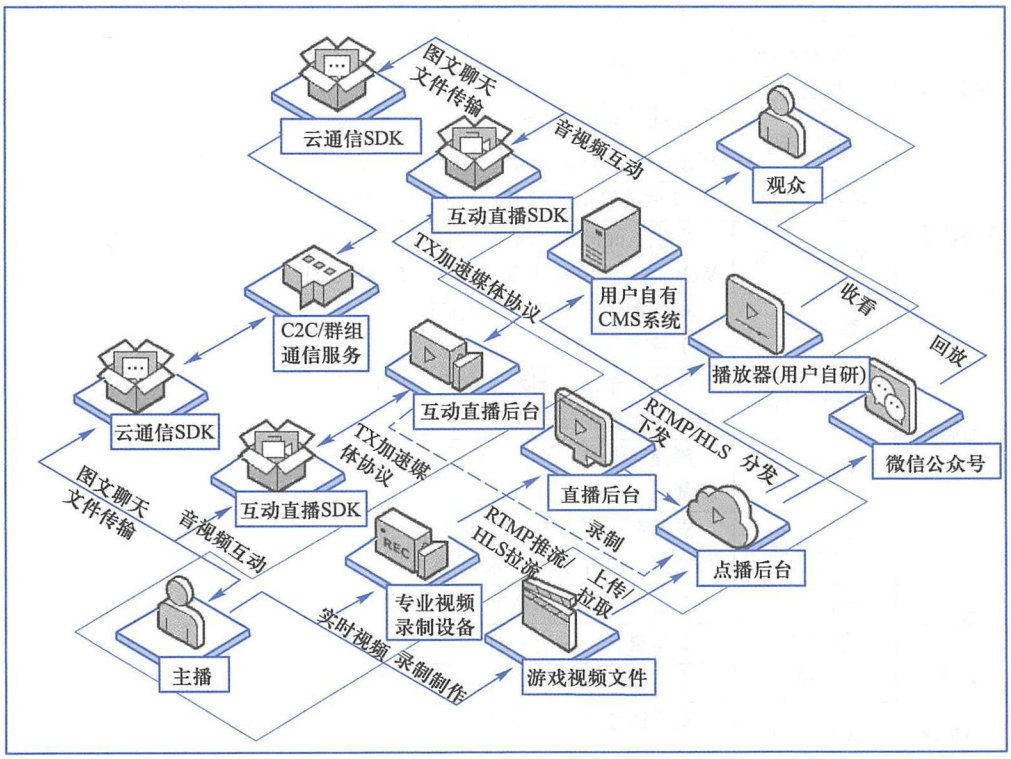


图 2-12
游戏直播系统架构

2）视频门户场景

主要特性如下。

- 借助超过 500+节点的直播发布平台，帮助视频门户用户具备和腾讯视频底层服务水平一致的发布能力。
- 系统可轻松接入专业视频录制设备、播控平台等任意标准的 HLS/RTMP 直播源，快速发布演唱会、综艺节目、体育赛事、企业活动等各种直播内容。

- 提供灵活的播放器，帮助视频门户通过移动应用、网页等各种展示方式，轻松触达最终。
- 详细的统计分析更可帮助服务商精准地把握用户行为特征。还可直接录制并对接点播平台，提供完善的视频服务。

系统构架如下。

系统架构如图 2-13 所示。

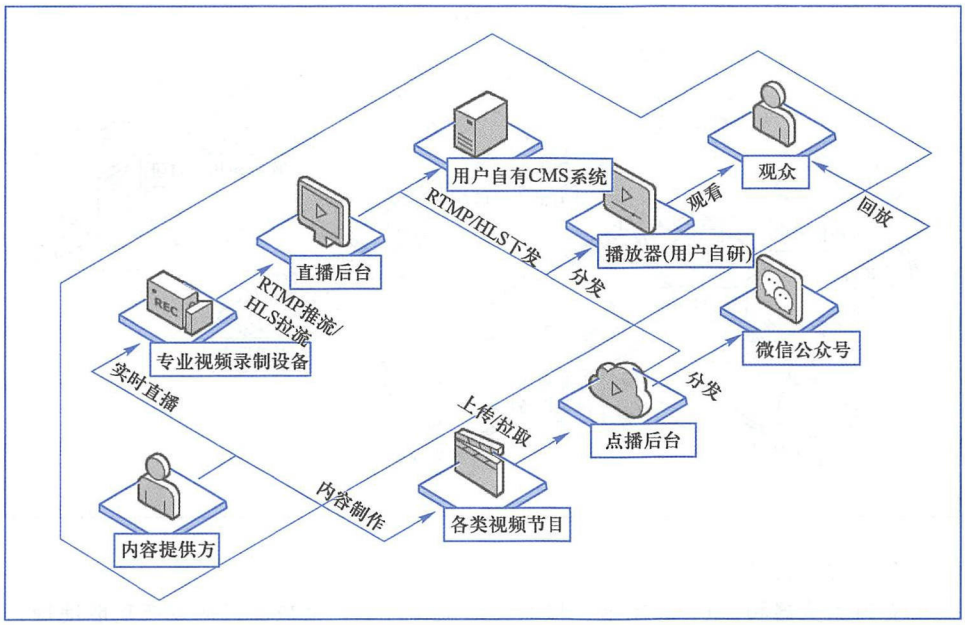


图 2-13
视频门户系统架构

3）社交直播场景

主要特性如下。

- 兼容各种平台和录制设备直播源，提供快速的接入方式。
- 提供灵活的播放器，帮助视频门户通过移动应用、网页等各种展示方式，轻松触达最终。
- 详细统计分析功能，帮助对 APP 或直播数据进行分析，提供完善的视频服务。

系统构架如下。

社交直播系统架构如图 2-14 所示。

2. 点播服务

（1）基本描述

腾讯云视频解决方案基于腾讯多年的技术积累与基础设施建设，为有视频应用相关需求的客户提供包括点播服务、音/视频通信服务在内的一站式解决方案。

目前，腾讯云视频解决方案面向腾讯云注册用户开放。

（2）业务介绍

① 极致加速体验。

遍布全国的 BGP 网络，覆盖 17 个以上运营商和 600 多个 CDN 视频加速节点。

② 强悍的转码能力。

超过 10 000 台分布式转码集群，2 000 并发转码。保障转码质量效率，可灵活添加水

印，并设定不同的转码格式，完美匹配不同的应用场景。

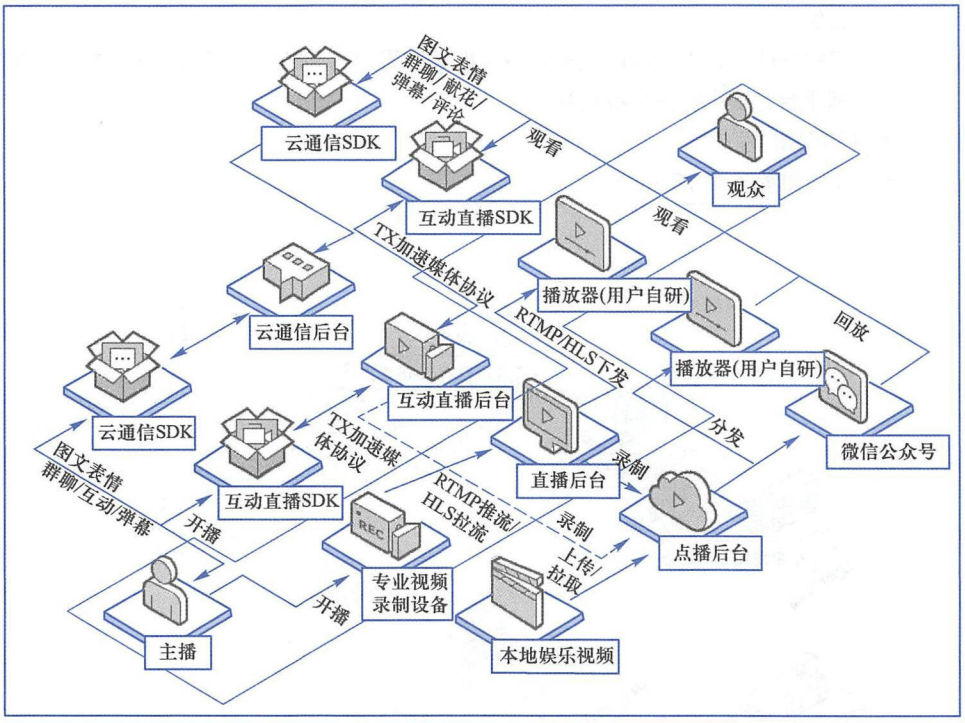


图 2-14
社交直播系统架构

③ 灵活上传发布。

支持浏览器和 API 等本地、拉取灵活上传方式，播放器代码发布功能便捷，Web/Android/iOS 播放器 SDK 灵活。

④ 免费的视频服务。

通过腾讯视频 V+认证，享受 H5 高清及原创保护等多项免费服务。

⑤ 视频管理。

可通过管理控制台或 API 对视频文件进行管理，包括分类、添加标签等操作，所有信息支持导出查看。

⑥ 视频分发。

可快速获得视频播放 URL 链接地址，以及自定义播放器代码，支持 CDN 快速分发。

⑦ 自定义播放器 SDK。

提供包括 Web/Android/iOS 播放器 SDK，帮助客户快速地构建自己的全平台高度定制化视频发布能力。

⑧ 安全功能。

提供包括黑白名单、播放密码等安全特性，更有防盗链功能，保护用户的视频内容万无一失。

⑨ 统计特性

提供丰富、细粒度的统计特性，包括时间维度等细节信息，快速查看视频发布数据，评估发布效果。

(3) 产品构架

典型的产品构架如图 2-15 所示。

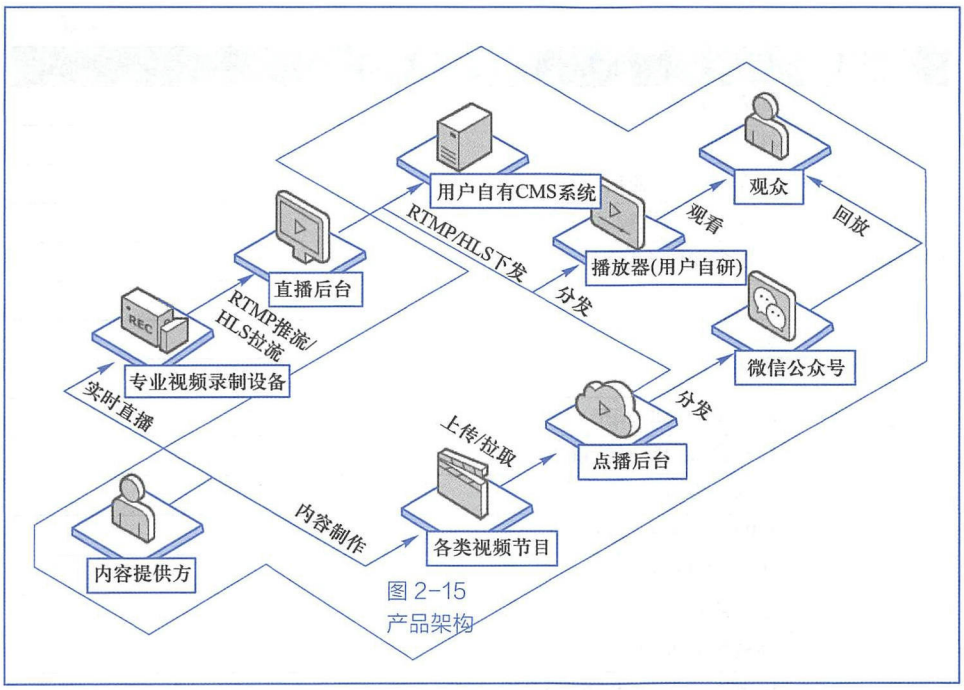


图 2-15
产品架构

图 2-15
产品架构

- 上传组件：用户通过上传、拉取等方式，将视频存放于点播后台，并根据需要进行自动转码。
- 管理组件：用户可通过 API、管理控制台方式，对文件进行管理操作和转码操作。
- 下发组件：通过遍布全国的 CDN 系统，为点播用户提供快速灵活的播放方式。用户可使用自研或腾讯云提供的播放器 SDK 方便同自有业务集成，也可对视频生成专属微信公众号链接，使用微信公众号发布。

（4）主要功能

点播产品支持功能如表 2-8 所示。

表 2-8
功能列表

类别	功能名称
视频上传	批量视频上传
	Web 页面上传
	断点续传
	SDK 上传
	大文件上传（支持 10 GB）
	文件多重备份
视频管理	视频信息编辑
	视频封面图片管理
	视频模糊搜索
	视频信息 CSV 导出
	视频多级分类及标签

笔 记

续表

类别	功能名称
视频管理	列表式视频管理
	在线预览
	动态 URL 防盗链
	微信公众号视频发布接口
	黑白名单防盗链
转码	流畅 240P 发布
	标清 480P 发布
	高清 720P 发布
	超高清 1080P 发布
	转码添加自定义位置水印
	不转码设置
	MP4 格式转码发布
	HLS 格式转码发布
播放器	播放器片头/暂停/片尾贴片
	播放器 LOGO 自定义
	Web/Android/iOS 播放 SDK
	渐进式加载视频文件
	iFrame 快速发布 Web 播放器
	播放器密码设置
	Android 智能电视播放 SDK
	Web 播放器弹幕展示接口
	分享至新浪微博和 QQ 空间等
	播放状态获取和事件设置 API
统计功能	空间统计
	流量统计
	视频播放情况
	请求数统计
	ISP 统计
	地域统计

（5）应用场景

点播系统可应用在各种丰富的场景，并可搭配腾讯云视频解决方案的其他产品，实现视频门户、在线教育、企业视频发布等各种场景。

1）视频门户

主要特性如下。

- 腾讯云点播服务为视频门户提供完善的视频服务。无论是已有海量视频资源，还是初涉在线视频服务，均可通过灵活的控制台或 API 方式上传拉取各类视频。
- 超万台转码集群提供强悍的转码能力，更快发布精彩视频。更有播放器密码、黑白名单、动态 URL 防盗链 3 大安全功能，保护用户精心制作的优质内容万无一失。
- 覆盖全国的 500+CDN 下发节点确保任意时间、任意地点的用户均可流畅观看视频。

更有完善的 API 接口，全方位对接用户自有 CMS 媒体内容管理平台。

系统构架如下。

视频门户系统架构如图 2-16 所示。

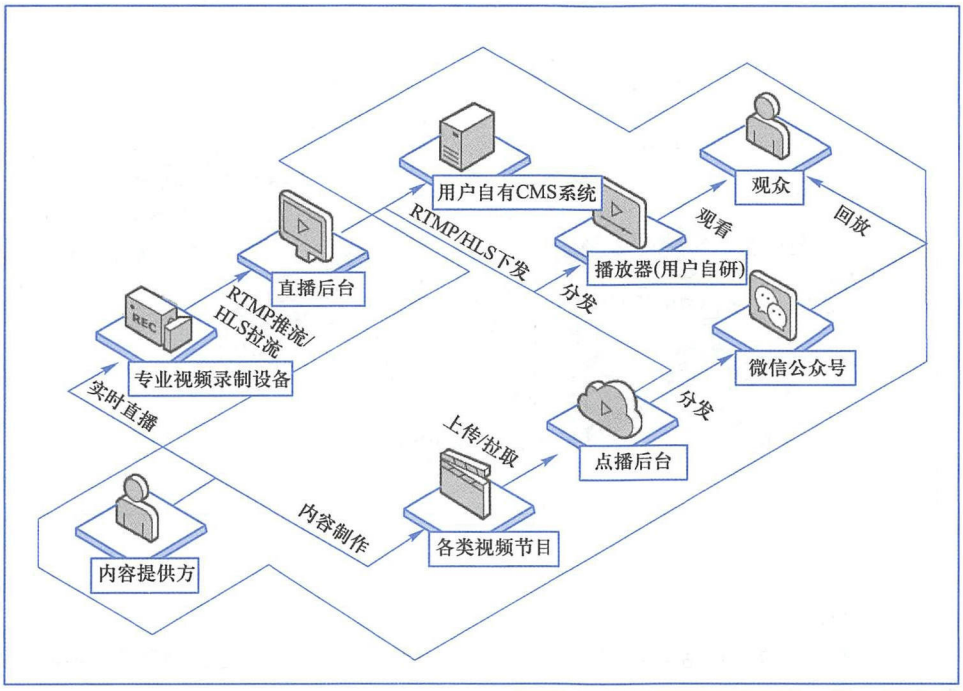


图 2-16
视频门户系统构架

2）在线教育

主要特性如下。

- 腾讯云点播服务为在线教育服务商提供便捷的一站式服务。
- 可通过灵活的控制台或 API 方式上传拉取教学视频，更有播放器密码、黑白名单、动态 URL 防盗链 3 大安全功能，保护用户精心制作的教学内容万无一失。
- 强悍的转码能力和覆盖全国的 500+CDN 下发节点确保任意时间、任意地点的学生用户均可快速查看教学内容。
- 更提供了全面开放的播放器接口，便于用户自定义限时播放、禁止拖拽等各类教学功能。

系统构架如下。

在线教育系统架构如图 2-17 所示。

3. 云通信服务

（1）基本描述

腾讯是国内最大也是最早的即时通信开发商，QQ 和微信已经成为每个互联网用户必不可少的应用程序。现在，腾讯将高并发、高可靠的即时通信能力进行开放，开发者可以很容易地根据腾讯提供的 SDK 将即时通信功能集成入 APP 中。

针对开发者不同阶段的需求及不同场景，云通信提供了一系列解决方案，包括：Android/iOS/Windows/Web 的 SDK 组件、服务端集成接口、第三方回调接口等。利用这些组件，可以在应用中构建自己的即时通信产品，解决开发者面临的高并发、高可用性的一系列问题。

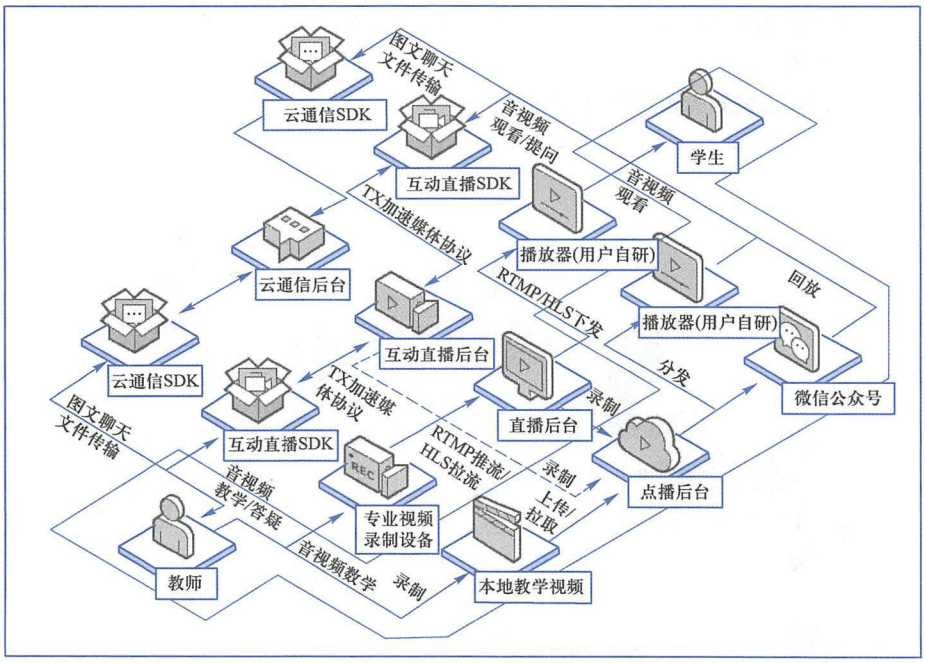


图 2-17
在线教育系统构架

（2）业务介绍

① 单聊。

1V1 聊天，提供包括文字、图片、语音、地理位置、文件、自定义消息等多种能力。除此之外，还提供 iPhonePush 消息提醒服务。

② 群聊。

多人聊天服务，内置公开群、私有群、聊天室、互动直播聊天室和在线成员广播大群 5 种群组形态，能够适应各种群组需求的场景。

- 公开群：具备创建者及管理员身份，拥有严格的准入制和管理权限。
- 私有群：只允许群成员邀请好友加入，非群成员无法搜索到该群，是一种较为私密的聊天环境。
- 聊天室：群成员可以随意进出，组织较为松散。成员可以看到进入聊天室之前的聊天消息。
- 互动直播聊天室：与聊天室类似，但群成员人数无上限，并且支持以游客身份接收聊天消息。
- 在线成员广播大群：成员人数无上限，支持以游客身份接收群消息，适用于向 APP 全体在线成员推送消息的场景。

群组具备高度可定制性，包括自定义群组形态、自定义字段、自定义群组 ID、自定义事件回调等。APP 可以根据自己的需求进行深度定制。

③ 资料关系链托管。

提供资料、关系链托管的一套整体解决方案，可存储用户的资料（如昵称、头像、自定义资料字段）、好友列表、黑名单等。云通信资料关系链托管服务提供高达 12 份的备份服务，多机房异地部署，提高服务质量及容灾效果。

④ 账号登录集成。

登录服务（Tencent Login Service，TLS）是腾讯为开发者快速完成账号集成接入云通

信服务而提供的一套通用账号登录组件，实现包括 QQ、微信、新浪微博、人人在内的多种账号类型登录使用音/视频通信和即时通信。开发者可以通过简单的 SDK 集成、便捷的接口调用，即可实现原本复杂的登录验证服务，使代码量急剧减少。

⑤ 接入服务系统。

接入服务为 APP 提供高连通、高可靠、强安全的网络连接通道的服务。自研多重最优寻址算法，具有全网调度能力，使用智能兼容技术穿透网关策略、长连接多路复用、传输层协议优化、通道加密等，让业务不必关注网络细节，即可安全与业务后台简单可靠地通信。

⑥ 消息推送服务。

消息推送服务是基于云通信 IM 的通信架构实现的一组 REST API，用以支持 APP 应用的全员推送、标签推送等消息推送需求，客户端可通过 SDK 在线推送、Android 后台通知和 APNS 接收推送的消息。

（3）架构介绍

云通信提供单聊、群聊、资料托管、关系链托管、账号托管全方位解决方案，并提供完善的 APP 接入、后台管理接口，如图 2-18 所示。

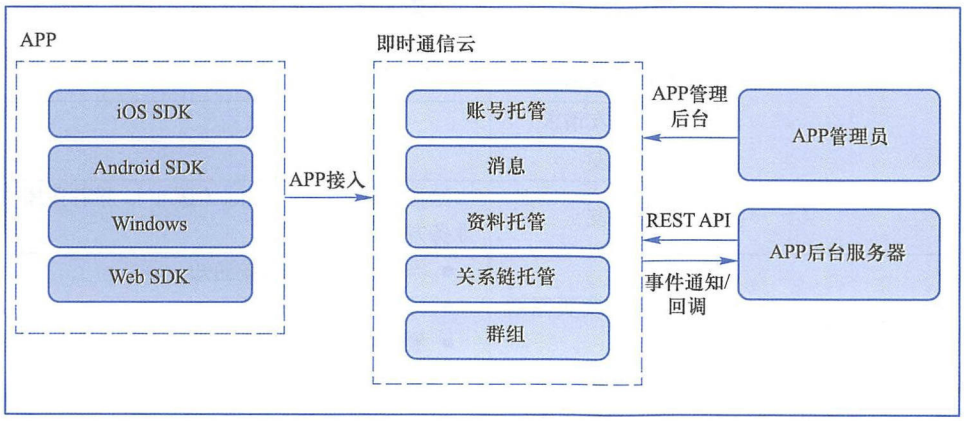


图 2-18
云通信架构

（4）主要功能

支持平台如表 2-9 所示。

功能	子功能	功能备注/常见问题
支持 Android/iOS/Windows C++/Web	—	支持 Android/iOS/Windows C++/Web 等主流平台互通，是跨终端全平台的服务

表 2-9
支持平台

账号集成如表 2-10 所示。

功能	子功能	功能备注/常见问题
独立模式	支持自有账号	
	支持第三方账号	支持 QQ、微信、新浪微博等第三方登录账号
	匿名	支持匿名/游客模式登录
托管模式	支持自有账号	—
	支持第三方账号	支持 QQ、微信、新浪微博等第三方登录账号
	匿名	支持匿名/游客模式登录

表 2-10
账号集成

支持会话类型如表 2-11 所示。

表 2-11
支持会话类型

功能	子功能	功能备注/常见问题
会话类型	单聊	
	群聊	支持公有群/聊天室/讨论组/直播聊天室
会话获取	—	发消息时首先需要先获取会话，获取会话需要指定会话类型（群组和单聊）
会话删除	—	此处仅删除会话记录，会话中的消息并未删除

关系链管理如表 2-12 所示。

表 2-12
关系链管理

功能	子功能	功能备注/常见问题
关系链管理	好友备注	
	获取好友列表	
	获取好友资料	
	查找好友	可通过用户 ID 或昵称查找好友
	添加好友	
	申请添加好友	需要选择默认是否需要申请理由，现在默认不需要
	同意好友请求	
	拒绝好友请求	
	删除好友	
	推荐好友	给推荐好友存储 buffer（缓冲），推荐逻辑业务方定义
关系链事件通知	关系链变更系统通知	• 添加/删除好友系统通知 • 好友申请系统通知 • 删除未决请求通知 • 好友资料变更系统通知
黑名单	加入/移除黑名单	
联系人管理	最近联系人	最近联系人漫游
		最近联系人删除
系统消息	支持在线状态的消息推送	系统消息不能由用户主动发送，是系统后台在相应的事件发生时产生的通知消息
离线消息	支持离线时消息推送	• 目前 iOS 支持 APNs 离线推送，Android 支持小米、华为推送 • 用户在登录状态（应用），在后台运行时，用户离线 7 天内可收到离线推送，离线大于 7 天则不能收到离线推送消息
消息推送	—	支持后台全员推送消息给用户

消息管理如表 2-13 所示。

表 2-13
消息管理

功能	子功能	功能备注/常见问题
消息状态	发送状态	发送成功/失败/发送中
	已读状态	用户阅读某个会话后，应用需上报消息已读，SDK 会设置该会话之前的所有消息为已读

续表

功能	子功能	功能备注/常见问题
消息缓存	SDK 会缓存消息内容（成功或失败都会缓存）	消息缓存本地
	对图片、语音等资源类消息，消息体只会包含描述信息，需调用接口下载数据	如有缓存本地需求，需应用研发支持
消息获取	异步获取消息	通过设置回调得到消息数据
	实时消息同步	通过回调接口实时同步消息
消息过滤	敏感词过滤	
消息删除	—	使用消息的去除（remove）方法可以在本地删除消息
未读计数	—	
消息漫游	—	• 保存 7 天消息 • 保存一定大小的消息，超出滚动
消息拓展属性	可通过接口获取用户部分资料	能从消息中获取的信息：暂支持通过消息获取用户 ID、用户昵称、用户头像
平滑迁移	—	支持自家和第三方合作的历史版本消息互通

群组管理如表 2-14 所示。

表 2-14 群组管理

功能	子功能	功能备注/常见问题
群组类型	支持私有群/公有群/聊天室/直播聊天室	• 私有群：允许普通成员修改群组资料、拉人进群，允许群主退出等 • 公有群：拥有更为严格的群组权限管理机制，类似 QQ 群 • 聊天室：相对于公有群，聊天室的历史消息可被查看，类似兴趣群 • 直播聊天室：针对直播场景中参与人数多和消息量大等特点，云通信提供了直播聊天室
群组管理	新建群组	—
	解散群组	群主可操作
	转让群组	只有群主才有权限进行群转让操作
	禁言	群主和管理员可对群普通成员做禁言操作
	群资料	对于公有群和聊天室，只有群主或者管理员可以修改；对于私有群，任何人可修改
	群简介	
	群公告	
	群成员资料	只有群主或者管理员可以对群成员的身份进行修改，如设置管理员
	修改群名片	—
	更多功能权限管理	详情请参见“群组功能权限介绍”
用户操作	拉用户入群	此操作只对私有群有效
	删除群组成员（带原因）	对于私有群，只有创建者可删除群组成员；对于公有群和聊天室，只有管理员和群主可以踢人

续表

功能	子功能	功能备注/常见问题
用户操作	邀请入群	—
	邀请入群同意/拒绝消息	—
	用户主动申请加入群组	此操作只对公有群和聊天室有效
	申请加群同意/拒绝消息	—
	设置/取消管理员	—
	退出群组	对于私有群，全员可退出群组；对于公有群和聊天室，群主不能退出
群组查找	通过群 ID 搜索群	—
	通过群名称搜索群	—
群消息	普通消息	请参见“消息管理”内容
	系统消息	• 消息类型：邀请加入群/用户退群/群名称修改 • 离线存储时间：7 天 • 离线数量上限：100 条
群参数上限	群数规模	一个业务最多支持 1 亿群组
	单群成员数	单个群组成员数上限为 1 万（注：直播聊天室加入人数无上限）
	单用户加入的群个数	每个用户加群上限为 5 千
群资料托管	群资料托管空间	—
	群成员资料托管空间	—
获取信息	获取群组成员列表	—
	获取群组成员入群时间	—
	获取加入群组列表	—
	获取群组资料	群组资料字段列表：群名/公告/简介/创建人/创建时间/群成员数量（备注：暂不支持自定义字段）
	获取群组公开资料	当用户不在群组时，需要通过接口获取群的公开资料

群组功能与权限介绍如表 2-15 所示。

表 2-15
群组功能与权限介绍

类型	控制维度	讨论组	公开群	聊天室	直播聊天室
群基础资料	是否允许普通成员修改资料	是	否	否	否
	是否允许被搜索到	否	是	是	否
	是否允许需申请加群	否	是	是	否
群关系链	是否允许直接拉人入群	是	否	否	否
	是否允许群主退群	是	否	否	否

续表

类型	控制维度	讨论组	公开群	聊天室	直播聊天室
群关系链	是否允许群主解散群	否	是	是	否
	是否允许设置管理员	否	是	是	否
	是否允许普通成员踢人	否	否	否	否
群消息	无限制历史消息	否	否	是	是
	成员变更通知	是	是	否	是
	是否在首条消息之后激活群	是	否	否	否
群功能	是否设置群成员人数上限	是	是	是	否
	是否直接支持不登录从 H5 接收消息	否	否	否	是

控制台如表 2-16 所示。

表 2-16 控制台

功能	子功能	功能备注/常见问题
Crash 上报		可通过控制台查看分析 Crash 数据
回调自助配置		可通过控制台快捷配置回调 URL 和数十种需求的回调

（5）应用场景

即时通信社交已经成为刚需，大量基于移动设备的社交强弱关系应用发展快速，各种关系链社交、协同办公、电子商务、远程会诊、智能硬件等垂直领域都有强烈的即时通信需求。

4. 云存储

（1）基本描述

云存储（Cloud Redis Store，CRS）是腾讯云基于分布式缓存领域多年技术沉淀和 Redis 类业务运营需求，打造的一款高可用、高可靠的 Redis 服务平台。支持分布式存储摆脱单机的性能和容量限制，从而提高 Redis 的服务能力；通过主从热备自动容灾提供高可用性；通过多个节点存储多份数据备份保证高可靠性。

CRS 支持主从版和集群版两种规格，提供主—从双节点实例。主从版容量为 1 GB ~ 60 GB，满足一般用户的容量和性能需求。集群版理论上无容量上限限制，但超过 300 GB 容量需提供支持，可满足大容量和高性能的需求。

在使用上，CRS 基本兼容 Redis 协议，支持字符串、链表、集合、有序集合、哈希表等多种数据类型。

CRS 特性介绍如下。

- 分布式：用户的存储分布在多台物理机上，彻底摆脱单机容量和资源限制。
- 高可用性：每个实例均提供主从热备、宕机自动监测、自动容灾。
- 高可靠性：数据持久化存储，可提供每日冷备和自助回档。
- 弹性扩容：集群版支持单实例无上限扩容，不用再担心资源受限。在扩容过程中不中断服务，真正做到用户无感知。
- 安全性：支持私有网络 VPC，提高缓存安全性。

（2）业务介绍

① 产品规格。

腾讯云 Redis 提供了两种版本的 Redis，分别是主从版和集群版，两者均采用双机热备、自动容灾架构。

主从版支持 1 GB ~ 60 GB 的任意规格，集群版最大 300 GB，理论上无上限。

② 自动容灾。

服务采用双机热备架构，主机故障后，服务秒级切换到备机，服务切换不影响线上业务，整个过程不需要做任何处理。节省了开发主备系统带来的人力成本和时间成本。

③ 弹性扩容。

控制台一键式扩容，扩容过程中无须停止服务，客户无须做任何处理。

④ 超高性能。

性能高达 10W+QPS，超高的性能可以满足绝大部分场景需求，非常适用于游戏、移动、广告、电商等开发场景。

⑤ 监控。

指标丰富，告警灵活，多达 30 余项的专业数据指标的监控，如出入网流量监控。提前预警风险，也能快速定位和解决问题。

（3）架构介绍

腾讯云 Redis 目前提供单机版和集群版两个版本供用户选择，区别详见表 2-17 所示。

表 2-17
腾讯云 Redis 类型

特性	单机	集群
容量	最大 60 GB	无上限
功能	全兼容原生 Redis 协议	遍历，事务，Lua 脚本受限
模型	单线程	多线程
value 适配	同原生 Redis	30 KB 内性能最佳
key 适配	同原生 Redis	上限 127 字节
连接数	可配置	暂无限，暂不可配置
超时时间	10 mins，可配置	30 mins，暂不可配置

注意

集群版单次请求的协议大小小于 1 MB。

选型推荐如下。

① 单实例比较小的开发者，例如游戏中存储单区单服数据量，建议使用单机版。

② 业务数据增长快，在突破单机 60G 上限的情况下，建议初始选择集群版。

（4）主要功能

① 数据存储的持久性。

数据存储的持久性是指在合同期内数据保存不丢的概率，即每月完好 key-value 数量/（每月完好 key-value 数量+每月丢失 key-value 数量）。以自然月为统计周期，不满一个月按一个月计。

云存储 Redis 的持久性为 99.99%。意味着每月用户存储 10 000 个 key，即每月只有

一个丢失的可能性。

② 数据可销毁性。

数据可销毁性是指用户要求删除云存储 Redis 表数据时，数据将从内存及磁盘中删除，并且原磁盘使用空间将会被覆盖，从而实现彻底删除，无法复原。

下线的存储服务器报废时，磁盘通过消磁销毁里面的数据。

③ 数据可迁移性。

云存储 Redis 提供表数据导入和导出功能。导入和导出的数据文件按照云存储 Redis 的备份文件格式组织。文件组织格式可以提供给用户。

④ 数据私密性。

云存储 Redis 通过鉴权中心防火墙机制，限制只能被同一云账号的服务器访问，从而隔离其他账号的非法访问。在无用户授权的情况下，腾讯云不能查看用户存储的数据，不能查看用户的操作流水。

⑤ 数据知情权。

用户对存储在云存储 Redis 的数据有以下知情权。

- 数据存储的数据中心位置（可以通过企业 QQ 咨询）。
- 数据备份数量及备份数据存储的数据中心位置（可以通过企业 QQ 咨询）。
- 用户有权选择本身数据的数据中心位置，无权选择备份数据的数据中心位置。
- 存储数据所在地与数据中心相关法律（可以通过企业 QQ 咨询）。
- 用户所有数据不会提供给任意第三方，除非政府监管部门监管审计需要。
- 云存储 Redis 不会分析用户存储的数据。

⑥ 数据可审查性。

腾讯云在依据现有法律法规体系下，出于配合政府监管部门的监管或安全取证调查等原因的需要，在符合流程和手续完备的情况下，可以提供云存储 Redis 相关信息，包括关键组件的运行日志、运维人员的操作记录、用户操作记录等信息。

⑦ 业务功能。

云存储 Redis 提供购买、数据增删改查、数据导入、数据导出、扩容、自助清理、查看监控信息等功能。所有功能均已提供详细的功能介绍和使用说明文档。每项会影响用户数据结果的功能变更均通过企业 QQ、电话或者邮件的方式通知用户。

⑧ 可用性。

腾讯云存储 Redis 承诺 99.95%的业务可用性，即单个数据库实例每个服务周期所有可用时间/服务周期不低于 99.95%。其中业务不可用的统计单元为用户单数据库实例。

业务故障的恢复正常时间在 5 分钟以下，不计入业务不可用性计算中，不可用时间指业务发生故障开始到恢复正常使用的时间，包括维护时间。

（5）应用场景

云存储 Redis 提供高可用、高可靠的键值数据库服务，不仅可以用在使用 Redis 协议的缓存场景中，也可以当作最终落地的 NoSQL 数据库使用。根据不同场景的需求，提供单机版和集群版服务。

① 单机版。

单机版对 Redis 开源协议有极好的兼容性，适用于对协议兼容性要求高、部分 Value 较大（大于 10 MB，如游戏里的全区全服用户排行榜）的场景，如图 2-19 所示。

笔记

② 集群版。

分布式服务，能提供高弹性的 Redis 存储服务，适用于对性能和容量要求高的场景，如图 2-20 所示。

图 2-19
单机版

图 2-20
集群版

5. 云服务器

(1) 基本描述

腾讯云云服务器（CVM）在云中提供可扩展的计算容量，避免了使用传统服务器时需要预估资源用量及前期投入的情况。通过使用腾讯云 CVM，可以在短时间内快速启动任意数量的云服务器并即时部署应用程序。腾讯云 CVM 支持用户自定义一切资源，如 CPU、内存、硬盘、网络、安全等，并可在访问量和负载等需求发生变化时轻松地调整它们。

(2) 业务介绍

① 弹性。

致力于打造业界最为弹性的云端服务器管理平台，提供以下能力。

- 基于云硬盘的云服务器即时提升/降低硬件配置（不区分包年、包月或按量计费类型）。
- 基于云硬盘的云服务器即时扩容磁盘（不区分包年、包月或按量计费类型）。
- 云服务器即时升级/降级带宽（仅按量计费类型，包年、包月类型支持即时升级带宽）。
- 云服务器切换带宽计费模式到流量计费模式（仅包年、包月类型）。
- 云服务器随时切换 Windows 与 Linux 系统（不区分包年、包月或按量计费类型）。
- 弹性 IP 支持绑定各种网络环境下的主机。

② 稳定。

主机服务可用性 99.95%，数据可靠性 99.99%～99.999999%。支持宕机迁移无感知、数据快照、自动告警等功能，为服务器保驾护航。

③ 多线接入。

提供覆盖超过 20 家网络服务提供商的 BGP 秒级切换功能，让不同地区不同 ISP 的用户享受相同的极速体验。

④ 安全。

腾讯云数据安全提供完善的备份及回滚机制保障数据安全性。

笔记

腾讯云主机安全是由腾讯专业安全团队倾力打造的，为云服务用户提供 DDoS 防护、DNS 劫持检测、入侵检测、漏洞扫描、网页木马检测、登录防护等安全服务。

腾讯云安全服务有如下特点。

- 全方位安全防护。

为云服务器提供一体化的安全服务，包括安全体检（如漏洞扫描、挂马检测、网站后门检测、端口安全检测等）和安全防御（如 DDoS 防护、入侵检测、访问控制来保证数据安全与用户隐私）。

- 实时告警定期分析。

7×24 小时的安全服务，第一时间发现漏洞，实时免费通知。

- 免费方便安全保障。

无须为云服务购买昂贵的安全设备，购买云服务即可免费享用云安全服务。一键开通，零部署，方便、简单。

- 专业团队，可靠保障。

云安全是由具备多年安全经验与历练的腾讯安全团队倾力打造的，为云服务用户提供的专业、安全的服务，值得信赖。

当前，腾讯云安全主要提供 DDoS 防护、DNS 劫持检测、入侵检测、漏洞扫描、登录防护等服务。

⑤ 易用。

提供控制台可视化、API 等多种控制方法，满足不同的需求。

（3）主要功能

① 多地域多可用区。

覆盖国内华南、华东、华北 3 个地域。海外节点覆盖香港、新加坡、多伦多、硅谷 4 个地域。在靠近用户的地域部署应用可获得较低的时延。

② 4 大机型配置。

标准型（适合中小型 Web 应用、中小型数据库）、高 IO 型（适合低时延，I/O 密集型应用）、内存型（适合需要大量的内存操作、查找和计算的应用）、计算型（适合大型游戏服务器和广告服务引擎、高性能计算及其他计算密集型应用程序）。

③ 第 II 代云服务器。

第 II 代云服务器采用 Intel Broadwell（v4）CPU、DDR4 内存，拥有更好的计算能力，全面搭配增强网络，包转发能力最高可达 30 W，整数和浮点运算的性能翻倍，整体计算能力更强。

④ 灵活调整配置

云服务器提供丰富的计算、网络和磁盘的选择和调整。提供不关机随时调整网络。支持关机状态下基于云盘调整硬件和磁盘扩容。

⑤ 三种计费模式。

提供包年、包月和按量计费三种计费模式。

⑥ 丰富的镜像。

公有镜像（多种 Linux 及 Windows 操作系统类型）、服务市场镜像（集成了不同应用环境的第三方）、自定义镜像（由用户通过镜像制作功能制作的镜像）。

⑦ 跨地域复制镜像。

为了方便利用腾讯云不同地理位置区域的优势，提供了将自定义镜像复制到其他腾讯云地域的功能。

⑧ 多种磁盘种类。

普通本地盘、SSD 本地盘、普通云盘、高性能云盘、SSD 云盘。

⑨ 自定义访问控制。

通过安全组和网络 ACL 自定义主机和网络的访问策略，灵活自由地为不同实例设定不同的防火墙。

（4）应用场景

① 小型 Web/APP 应用、企业官网等静态展示类网站，如图 2-21 所示。

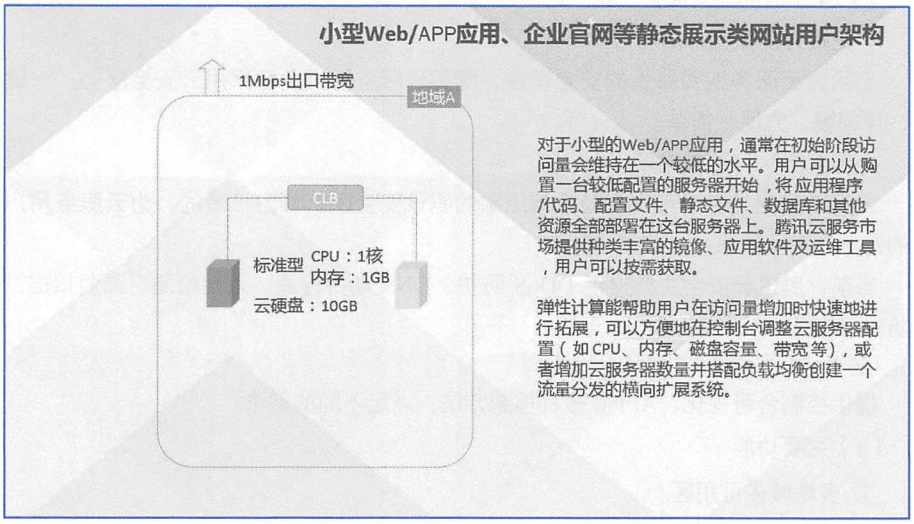


图 2-21
静态网站类

② 海量图片和视频等大文件流媒体应用，如图 2-22 所示。

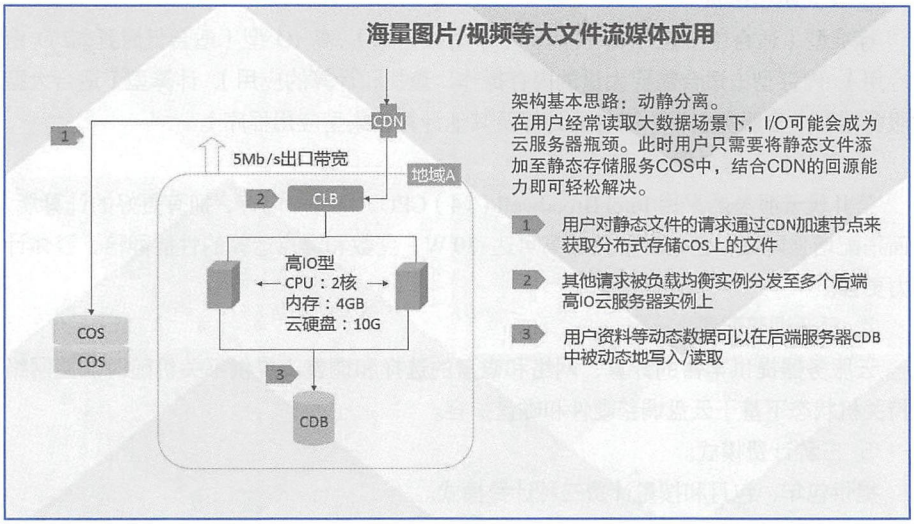


图 2-22
流媒体类

6. 云存储对象

(1) 基本描述

对象存储（Cloud Object Storage，COS）是腾讯云提供的一种存储海量文件的分布式存储服务，用户可通过网络随时存储和查看数据。腾讯云 COS 使所有用户都能使用具备高扩展性、低成本、可靠和安全的数据存储服务。

COS 通过控制台、API、SDK 等多样化方式简单、快速地接入，实现了海量数据存储和管理。通过 COS 可以进行多格式文件的上传、下载和管理。腾讯云提供了直观的 Web 管理界面，同时遍布全国范围的 CDN 节点可以对文件下载进行加速。

(2) 业务介绍

① 存储级别。

根据访问频度的高低，COS 提供了 3 种存储级别：标准存储、低频存储和近线存储。

注意

低频存储和近线存储将在 COS V5 版本上线。

② 标准存储。

标准存储为用户提供了高可靠性、高可用性和高性能的对象存储服务。标准存储拥有低访问时延和较高的吞吐量，因而适用于有大量热点文件、需要频繁访问数据的业务场景。

适用场景：热点视频、社交图片、移动应用、游戏程序、动态网站。

③ 低频存储。

低频存储为用户提供了高可靠性、较低存储成本和较低访问时延的对象存储服务。低频存储在降低存储价格的基础上，保持首字节访问时间在毫秒级，保证了用户在取回数据的场景下无须等待、高速读取。低频存储适用于较低访问频率的业务场景。

适用场景：网盘数据、大数据分析、政企业务数据、低频档案、监控数据。

④ 近线存储。

近线存储为用户提供了高可靠性、低成本的存储服务。近线存储是定位于客户在线存储和离线存储之间的数据存储。近线存储的最大特点是存储价格接近归档存储，但仍可以秒级响应。近线存储适用于数据访问频率极低但必要时仍需在线读取的业务场景。

适用场景：医疗存档、业务日志、合同法规归档、媒体资料存档、灾难备份。

(3) 服务等级协议

① 数据存储的持久性。

对象存储服务的持久性是指在合同期内存储的对象数据保存不丢的概率，即每月完好对象数量/（每月完好对象数量+每月丢失对象数量）。以自然月为统计周期，不满一个月按一个月计。

对象存储服务的持久性为 99.999999%。即用户每月每 100 000 000 个存储的对象数据，每月只有一个对象有数据丢失的可能性。

② 数据可销毁性。

数据可销毁性是指用户要求删除对象数据时，对象数据将从内存及磁盘中删除，并且原磁盘使用空间将会被新的对象数据覆盖，从而实现彻底删除，无法复原。下线的存储

笔记

服务器报废时，磁盘通过消磁销毁里面的数据。

③ 数据可迁移性。

对象存储服务存储的对象，可以通过HTTP的方式直接下载。用户可以用标准的HTTP的方式，很方便地把存储在腾讯云对象存储服务上的对象，转存到本地服务器或者其他云上。通过腾讯云提供的工具，还可以进行批量导入、导出对象。数据迁移可针对所有的文件类型。

④ 数据私密性。

对象存储数据的私密性，通过腾讯云安全平台提供的统一鉴权服务来保证。对象存储服务，通过 Bucket 的方式来组织对象数据。Bucket 中对象的创建、操作请求，都需要用 Bucket 所属用户的密钥计算得到一个签名，通过签名来校验请求的合法性。从而保证同一资源池中不同用户的对象数据互不可见。

Bucket 中对象数据的访问读取权限，可以由用户自由设置成公有读、私有读来控制。公有 Bucket 的对象的访问，不需要签名；私有 Bucket 的对象的访问，需要签名校验。

⑤ 数据知情权。

目前用户云对象存储部署在 3 大数据中心，分别是上海数据中心、深圳数据中心和天津数据中心。

数据备份数量及备份数据存储的数据中心位置，帮助用户选择网络条件合适的数据中心存储数据；冷备则是根据资源利用情况动态分配，用户默认无须选择数据中心和冷备中心位置，如果需要选择，可以通过提交工单帮助处理。数据中心要遵守当地的法律和中华人民共和国相关法律。用户存储的所有数据不会提供给任意第三方，除政府监管部门监管审计需要。

对象存储服务不会分析用户存储的数据。

⑥ 数据可审查性。

腾讯云在依据现有法律法规体系下，出于配合政府监管部门的监管或安全取证调查等原因的需要，在符合流程和手续完备的情况下，可以提供云数据库相关信息，包括关键组件的运行日志、运维人员的操作记录和用户操作记录等信息。

⑦ 业务功能。

腾讯云提供的对象存储服务功能，具体包括 Bucket 的创建、删除、访问权限控制、自定义访问域名、查看监控信息、运营数据，目录的创建、删除、属性更新、属性查看、文件列表、前缀搜索，对象的普通上传、分片上传、删除、属性更新、属性查看、下载等功能。所有功能均已提供详细的功能介绍和使用说明文档，与此同时，腾讯云还提供相应的 API 和 SDK，供用户使用。

⑧ 业务可用性。

对象存储服务承诺 99.95%的业务可用性，即用户每月业务可用时间应为 30 天×24 小时×60 分钟×99.95%=43 178.4 分钟，即存在 43 200−43 178.4=21.6 分钟的不可用时间，其中业务不可用的统计单元为用户单业务实例，对于对象存储服务而言，用户单业务实例即为 Bucket。

业务故障的恢复正常时间为 5 分钟及以下，不计入业务不可用性计算中。不可用时间指业务发生故障开始到恢复正常使用的时间，包括维护时间。

⑨ 业务资源调配能力。

对象存储服务，通过监控用户的存储、请求总量，自动地进行动态的扩容、缩容操

作，用户完全没有感知。资源扩展针对存储容量，单次扩展资源在 1 小时内可完成，每次扩容最小集合是 400 TB，最大不限。

⑩ 故障恢复能力。

腾讯云对象存储服务采用双副本数据冗余，即同一数据存储 3 份。至多两份数据损坏，都能保证客户服务的连续性。同时，腾讯云提供专业团队 7×24 小时帮助维护，能够及时修复出现故障的存储服务器。

（4）主要功能

COS 主要提供的功能如表 2-18 所示。

表 2-18 COS 功能

功能	描述
创建存储桶	在上传对象到 CO 之前，需要先创建存储桶
删除存储桶	如删除存储桶，需先删除存储桶中的所有对象和目录
设置存储桶访问权限	COS 提供权限控制（Access Control List，ACL），可在创建存储桶的时候设置相应的 ACL 权限控制，也可在创建之后修改访问权限
防盗链设置	COS 是按使用量收费的服务，为了减少存储在 COS 的数据被其他人盗链而产生额外费用，COS 支持通过防盗链设置来进行安全防护
回源设置	设置回源地址对获取数据的请求以多种方式进行回源读取，满足数据热迁移、特定请求的重定向等需求
托管静态网站	将存储桶配置成静态网站托管模式，并通过存储桶域名访问该静态网站
设置跨域访问	COS 提供 HTML 5 标准中的跨域访问设置，帮助实现跨域访问。针对跨域访问，COS 支持响应 OPTIONS 请求，并根据开发者设定的规则向浏览器返回具体设置的规则
域名管理	通过域名管理，可将自定义域名绑定到 COS 访问域名上，实现自定义域名访问存储桶下的对象，也可同时一键配置腾讯云 CDN（Content Delivery Network，内容分发网络）实现加速功能
上传对象	可上传对象（如文本文件、图片、视频、应用等多格式文件）到存储桶中
下载对象	可通过多种方式下载对象
搜索对象	可在存储桶或文件夹中搜索具有相同名称前缀的对象
设置对象权限	对象访问权限提供了基于对象维度的访问权限控制，且该配置优先级高于存储桶权限
设置 HTTP 头部	设置单个对象的 HTTP 头部
删除对象	单个或批量删除对象
创建文件夹	文件夹便于用户管理存储于 COS 的数据
删除文件夹	删除单个文件夹
API 文档	提供 COS 支持的 API 操作和相关示例
SDK 文档	提供主流语言 SDK 的开发操作和相关示例

笔 记

（5）应用场景

① 多点上传的 UGC 场景。

面对有大量 UGC 内容生成的业务场景，对象存储服务 COS 的大容量 IO 吞吐能力，使得 COS 相比传统的模式有效地解决了多点并发上传的问题，COS 自动选择就近节点上传，大幅度缩短了终端上传图片和文件的排队时间，同时也提高了文件上传的成功率。

② 频繁 IO 的网盘场景。

面对上传和下载比较频繁的网盘场景，对象存储服务 COS 相比原有的 NAS 设备，有更好的扩展性，COS 能够随着用户数据量的提高自动扩容；对于并发访问的情况，COS 比传统的 NAS 设备有更充足的带宽支持，避免了访问时延长或者服务不可用的情况出现。

③ 海量数据归档和备份的场景。

面对冷数据存储的场景，COS 提供了一套分级存储的方案，针对那些访问频率比较低，同时对访问速度要求不高的数据，建议将数据保存在低频存储中，如此可以在不降低数据耐久性的前提之下，降低存储费用约 40%，为保存档案和备份资料提供了一个海量低价的空间。

④ 热点资源的分发下载。

面对视频点播源、游戏资源等热点文件下发的场景，COS 结合 CDN 使用，灵活应对大流量和高并发的业务场景。可以使用 COS 作为源站，将热点资源放在 COS 中，然后通过 CDN 将资源下发给终端用户。这样一来，降低了下发的流量费用，降低了终端访问的时延，同时腾讯拥有强大的带宽，丝毫不用考虑由于流量过大而导致业务无法访问。

7. 云数据库

（1）基本描述

云数据库 MySQL（Cloud Database for MySQL）是腾讯云基于全球最受欢迎的开源数据库 MySQL 专业打造的高性能分布式数据存储服务，100%兼容 MySQL 协议，适用于面向关系型数据库的场景。

① 特点。

云数据库 MySQL 主要具有以下一些特点。

- 云存储服务，是腾讯云平台提供的面向互联网应用的数据存储服务。
- 完全兼容 MySQL 协议，适用于面向表结构的场景，只要是适用 MySQL 的地方，都可以使用云数据库。
- 提供了高性能、高可靠、易用、便捷的 MySQL 集群服务。
- 整合了备份、扩容、迁移等功能，同时提供 phpMyAdmin 管理后台，用户可以方便地进行数据库的管理。

② 术语。

- 常规实例：不含只读实例的实例被称为常规实例。常规实例可读、可写，在实例列表可见。MySQL 5.6 版本开启 GTID 后，常规实例可通过购买只读实例从而升级为主实例。
- 主实例：含有只读实例的实例被称为主实例。主实例可读、可写，在实例列表可见。在主实例的实例详情页中可对其所属的只读实例进行管理。

非卖品，仅供非商业用途或交流学习使用

笔记

- 只读实例：仅提供读功能的实例被称为只读实例。只读实例无法单独存在，必须隶属于某个主实例，唯一数据来源是从主实例同步数据，在实例列表中不可见。

（2）业务介绍

① 高弹性。

- 强大的硬件提供性能保障：PCI-E SSD 的强大 IO 性能保障数据库的访问能力。单实例最大支持 37 000 QPS、1 TB 存储空间。
- 提供灵活的计费方式：提供包年、包月费的计费模式。可以根据业务需求一次性支付一个月或几个月的数据库服务费用，避免一次性投入大量资金建设基础设备。
- 支持读写分离：CDB for MySQL 支持挂载只读实例，支持一主多从架构，轻松应对业务海量请求压力；支持带有负载均衡功能的 RO 组，大大简化从机之间压力分配的过程。

② 安全性。

- DDoS 防护：在用户遭到 DDoS 攻击时，能帮助用户抵御各种攻击流量，保证业务的正常运行。
- 数据库攻击防护：高效防御 SQL 注入、暴力破解等数据库攻击行为。
- 存储高可靠：提供在线的主从两份数据存储，确保线上数据安全。同时通过备份机制保存多天的备份数据，以便在灾难情况进行数据恢复。

③ 可用性。

- 实时热备：实时双机热备，3 日内无损恢复，5 日冷备数据 dump。
- 自动容灾：提供宕机自动检测和故障自动迁移。主备切换和故障迁移过程对用户透明。

④ 易用性。

- 轻松管理海量数据库：提供命令行和 Web 两种方式管理云数据库，并支持批量数据库的管理、权限设置和 SQL 导入。
- 数据导入与备份回档：提供多种数据导入途径完成初始化。每日自动备份数据，云数据库根据备份文件提供 3 天内任意时间点回档。
- 专业的监控与告警：多维度监控，自定义资源阈值告警，提供慢查询分析报告和 SQL 完整运行报告下载。
- 多种接入方式：支持外网访问和 VPC 网络，可通过这些接入方式将云数据库与 IDC、私有云或其他计算资源互联，轻松应用于混合云环境。

（3）服务等级协议

① 数据存储的持久性。

合同期内承诺每月用户申请实例的数据存储的持久性为 99.999 6%。即用户每月每 1 000 000 个实例存储的文件，每月只有 4 个实例有数据丢失的可能性。

② 数据可销毁性。

在用户要求删除数据或设备在弃置、转售前，腾讯云将采取磁盘低级格式化操作，彻底删除用户所有数据，并且无法复原，硬盘到期报废时将进行消磁。

③ 数据可迁移性。

腾讯云将提供标准的 MySQL 数据库文件格式的数据服务，并支持用户通过腾讯云提

非卖品，仅供非商业用途或交流学习使用

笔 记

供的导入/导出工具转存为标准的 SQL 文件，方便用户迁入云数据库，或者迁出到用户的虚拟主机或者本地服务器上。

④ 数据私密性。

腾讯云通过配置防火墙策略，采用白名单过滤机制进行网络隔离，通过数据库实例的用户名、密码和关闭数据库实例的权限控制机制，来保证同一资源池用户数据互不可见。

⑤ 数据知情权。

- 数据存储的数据中心位置。
- 数据备份数量及备份数据存储的数据中心位置。
- 帮助用户选择网络条件合适的数据中心存储数据，冷备则是根据资源利用情况动态分配，用户默认无须选择数据中心和冷备中心位置。
- 数据中心要遵守当地的法律和中华人民共和国相关法律。
- 用户所有数据不会提供给任意第三方，除政府监管部门监管审计需要。用户的行为日志会用于数据库运行状态的数据分析，但不会对外呈现用户个人信息数据。

⑥ 数据可审查性。

腾讯云依据现有法律法规体系，出于配合政府监管部门的监管或安全取证调查等原因的需要，在符合流程和手续完备的情况下，可以提供云数据库相关信息，包括关键组件的运行日志、运维人员的操作记录和用户操作记录等信息。

⑦ 业务功能。

腾讯云提供的云数据库实例的服务功能，具体包括 CDB 实例申请、登录、数据库增删改查、冷备提取、数据回档、数据导入、多线程数据导入/导出工具、运维数据查询 AP、密码重置。所有功能均已提供详细的功能介绍和使用说明文档。每项会影响用户数据结果的功能变更均通过站内信通知用户。

⑧ 业务可用性。

- 云数据库承诺 99.95%的业务可用性，即用户每月业务可用时间应为 30 天×24 小时×60 分钟×99.95%=43 178.4 分钟，即存在 43 200 − 43 178.4=21.6 分钟的不可用时间，其中业务不可用的统计单元为用户单业务实例。
- 业务故障的恢复正常时间为 5 分钟以下的，不计入业务不可用性计算中。不可用时间指业务发生故障开始到恢复正常使用的时间，包括维护时间。

⑨ 业务资源调配能力。

腾讯云承诺用户，申请计算资源扩容时，现有资源 50%以下容量，并且扩容实例资源小于 10 个，1 小时内完成；申请资源小于 30 个，24 小时内完成；多于 30 个，请提交工单咨询完成时间。每次单用户最大可扩展 100%最小 25 GB 的容量。

⑩ 故障恢复能力。

腾讯云提供专业团队 7×24 小时帮助维护。

(4) 应用场景

① 游戏应用典型场景。

CDB for MySQL 在游戏中作为服务端最终数据落地存储介质，其极高的性能足以应对大多数业务场景，如图 2-23 所示。

非卖品，仅供非商业用途或交流学习使用

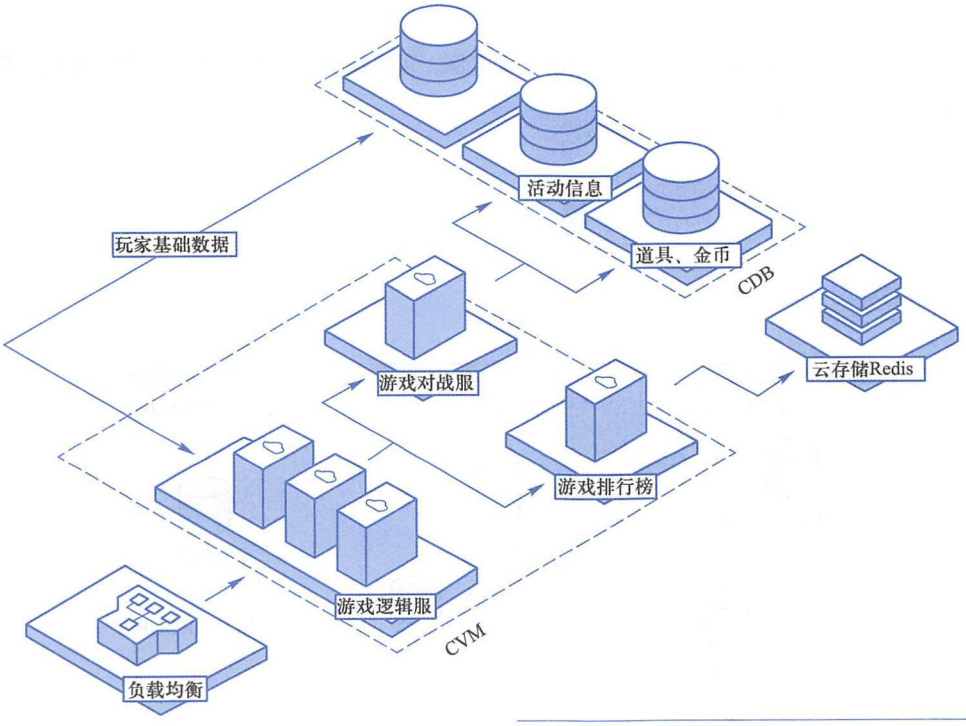


图 2-23
游戏应用典型场景

② 互联网/移动 APP 应用典型场景。

CDB for MySQL 在互联网/移动 APP 中作为服务端最终数据落地存储介质，针对行业读多写少的场景，可将热点库增加只读实例，大幅提升读取能力，如图 2-24 所示。

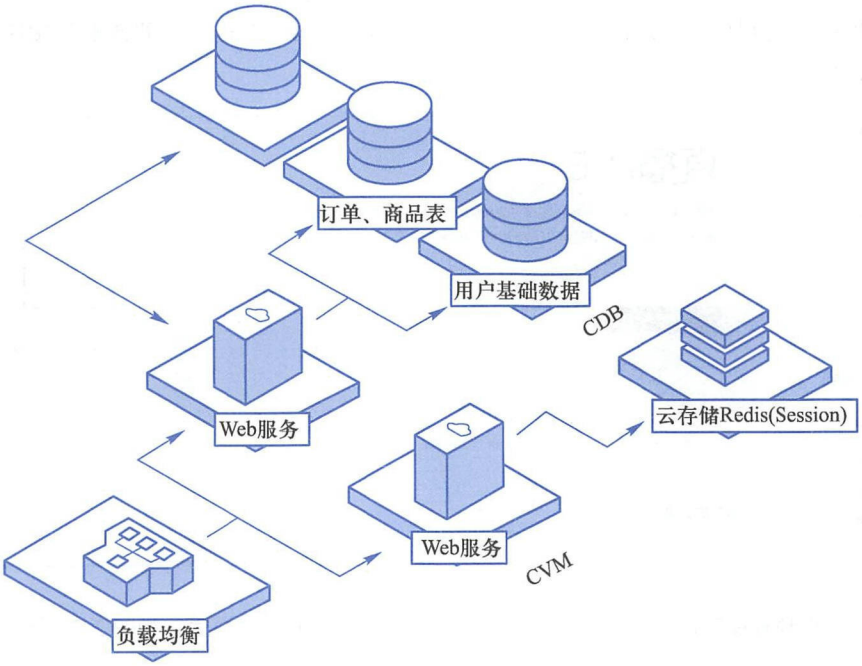


图 2-24
互联网/移动 APP 应用典型场景

非卖品，仅供非商业用途或交流学习使用

③ 读写分离场景。

CDB for MySQL 支持带 RO 组的只读实例，可将读负载分摊到各个只读实例上，保证服务的高可用，如图 2-25 所示。

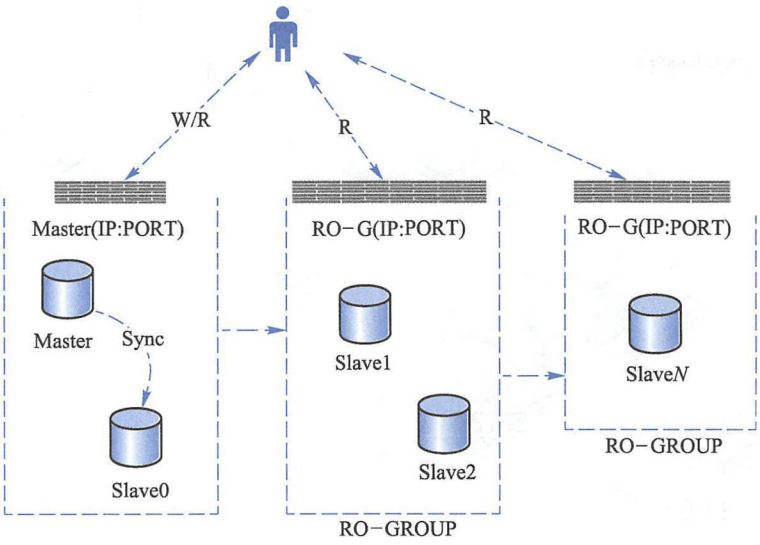


图 2-25
读写分离场景

任务实现

微课 2-2
服务租赁

1. 申请直播服务

打开腾讯云主页 https://cloud.tencent.com/并登录，在导航栏中选择“云产品→视频服务→直播”菜单命令，打开“直播 LVB”界面。单击“立即选购”按钮，如图 2-26 所示。

图 2-26
“直播 LVB”界面

在视频直播管理页面，单击“申请开通”按钮，打开申请页面，如图 2-27 所示。

非卖品，仅供非商业用途或交流学习使用

图 2-27
申请开通

按照要求及提示填写用户信息和业务信息（注：企业开发者可申请企业类型，个人开发者可申请个人类型）。单击“提交申请”按钮，等待腾讯云管理人员开通服务，如图 2-28 所示。

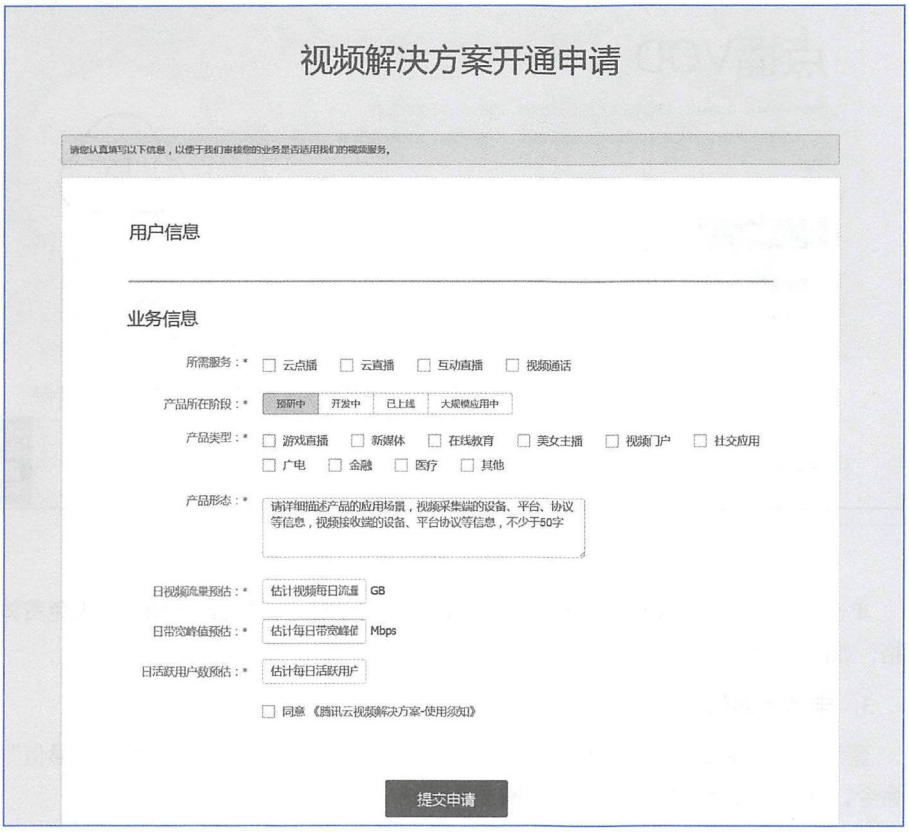


图 2-28
提交申请

非卖品，仅供非商业用途或交流学习使用

成功开通直播服务后，单击“视频直播→接入管理→直播码接入（推荐）”按钮，进入直播管理界面，可以看到自己的 appid、bizid 及应用信息等，在之后的开发过程中需要用到，回调 URL 需要自己配置，与后期自己部署的项目保持一致，如图 2-29 所示。

图 2-29
直播配置

2. 申请点播服务

登录腾讯云主页，在导航栏中选择“云产品→视频服务→点播”菜单命令，打开点播界面，如图 2-30 所示。

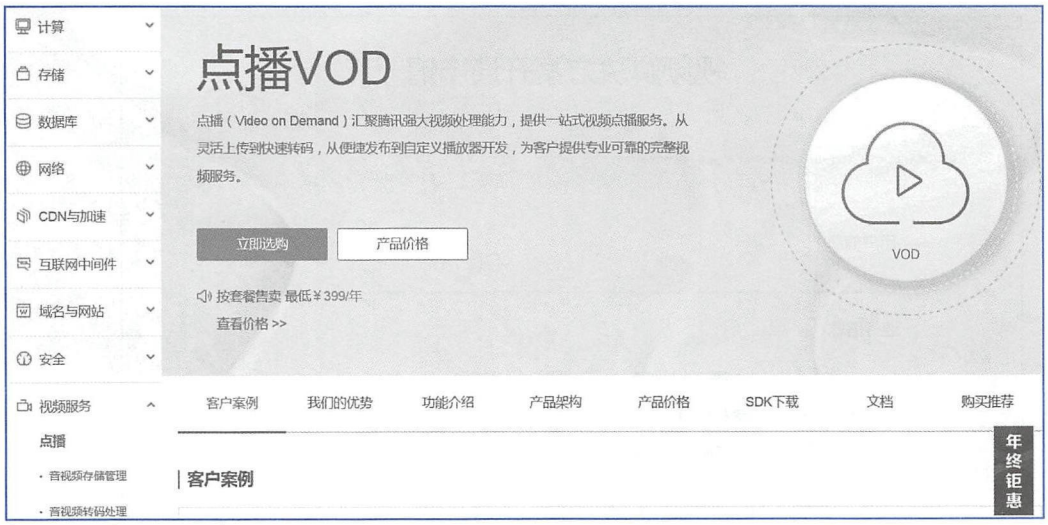


图 2-30
选购点播

单击“立即选购”按钮即可开通（请先通过实名认证），系统自动赠送 7 天免费体验资格，如图 2-31 所示。

3. 申请云通信

登录腾讯云主页，在导航栏中选择“云产品→基础产品→移动与通信→云通信”菜单命令，打开“云通信 IM”界面，如图 2-32 所示。

非卖品，仅供非商业用途或交流学习使用

图 2-31
点播控制台

图 2-32
“云通信 IM”界面

单击“立即使用”按钮，可以跳转到云通信管理界面，单击“申请开通互动直播”按钮申请开通云通信服务，如图 2-33 所示。

4. 申请云存储 Redis

在导航栏中选择“云产品→基础产品→数据库→弹性缓存（云存储 Redis）”菜单命令，打开“云存储 Redis”界面，如图 2-34 所示。

非卖品，仅供非商业用途或交流学习使用

图 2-33
开通云通信

图 2-34
“云存储 Redis”
界面

单击“立即选购”按钮，打开“云存储 Redis”的配置界面，按需求选择配置，开通即可使用，如图 2-35 所示。

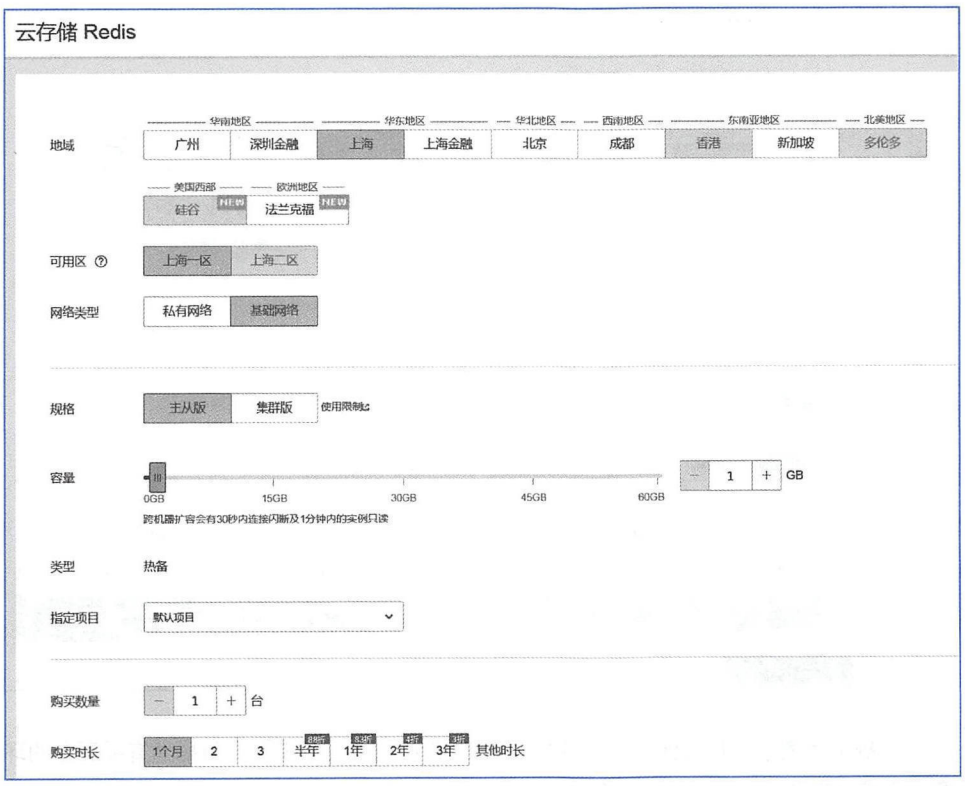


图 2-35
云存储配置

5. 申请云服务器

在导航栏中选择“云产品→基础产品→云计算与网络→云服务器”菜单命令，打开“云服务器 CVM”界面，如图 2-36 所示。

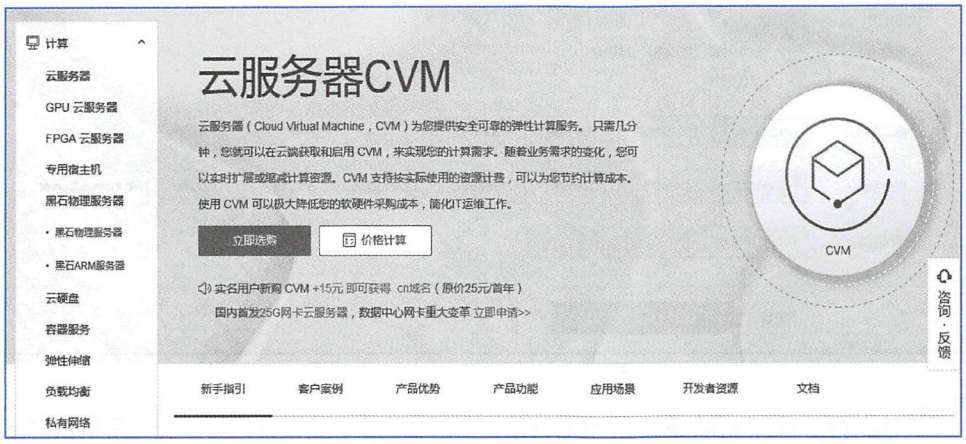


图 2-36
选购服务器

单击“立即选购”按钮，打开配置界面，根据自己的地理位置，选择距离自己位置较近地域的服务器，完成后单击“下一步：选择镜像”按钮，如图 2-37 所示。

版权所有，严禁传播，违者自负法律责任！

图 2-37
地域与机型配置

根据项目需求选择镜像，一般选择“CentOS 6.5 64 位”镜像作为搭建直播后台的环境，完成后单击“下一步：选择存储与网络”按钮，如图 2-38 所示。

图 2-38
选择镜像

根据自己的实际需求配置存储与网络，一般情况下保持默认即可，完成后单击“下一步：设置信息”按钮，如图 2-39 所示。

设置云主机的主机名，以及 root 用户的密码，用来登录云主机。安全组可以自己配置，本任务选择放通全部端口，最后单击“立即购买”按钮，如图 2-40 所示。

版权所有，严禁传播，违者自负法律责任！

图 2-39
选择存储与网络

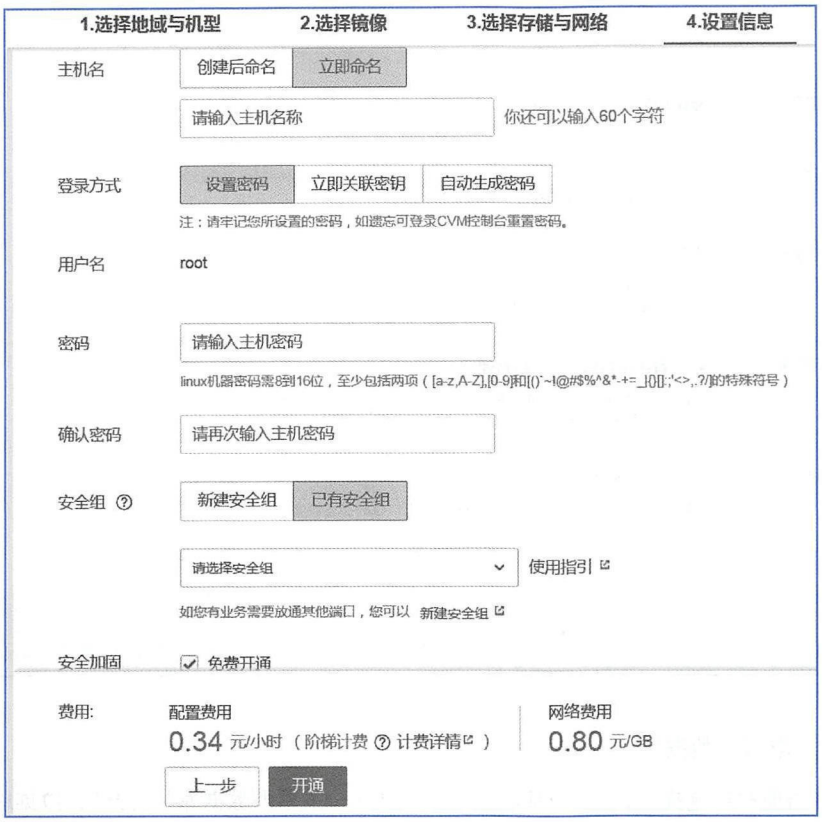


图 2-40
设置信息

版权所有，严禁传播，违者自负法律责任！

6. 申请对象存储

在导航栏中选择“云产品→基础产品→存储→对象存储”菜单命令，打开“对象存储 COS”界面，如图 2-41 所示。

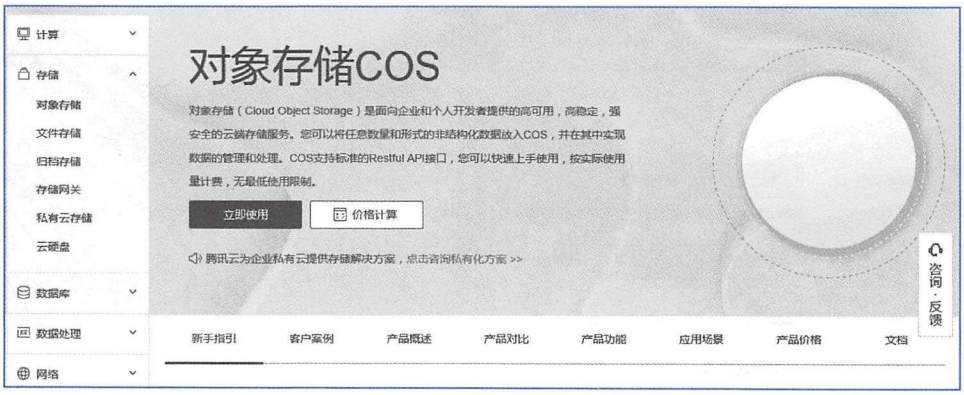


图 2-41
选购对象存储

单击“立即使用”按钮，打开配置页面。选择“云对象存储 v4→Bucket 列表”菜单命令，在 Bucket 列表中，单击“+创建 Bucket”按钮，新建 Bucket。一般情况选择默认配置即可，如图 2-42 所示。

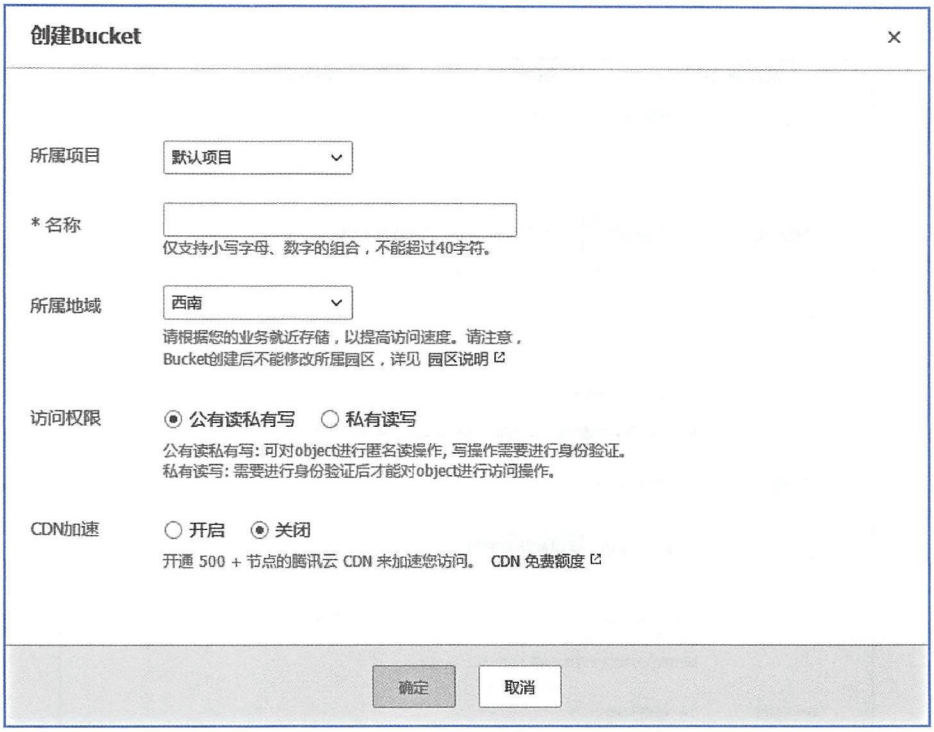


图 2-42
创建 Bucket

7. 申请云数据库

在导航栏中选择“云产品→基础产品→数据库→关系型数据库”，打开“云数据库 CDB for MySQL”界面，如图 2-43 所示。

版权所有，严禁传播，违者自负法律责任！

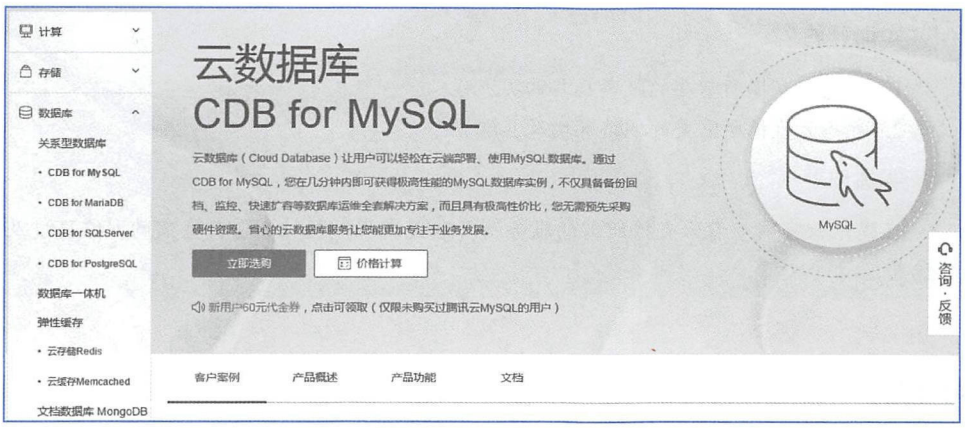


图 2-43
选购云数据库

单击“立即选购”按钮，打开配置界面，根据自己的需求配置开通即可，如图 2-44 所示。

图 2-44
配置云数据库

项目实训

实训题目

完成整套云直播所需要网络环境的配置。

版权所有，严禁传播，违者自负法律责任！

实训目的

1. 熟悉云直播所需要的服务及其购买流程。
2. 熟悉云直播所需要的网络环境及其配置。

实训内容

在腾讯云官网查看云直播所需的服务及其购买流程，同时查看云直播的网络环境及其配置。

单元小结

本单元讲解了如何对云直播系统进行设计及后台开发所涉及的 3 个框架（Bootstrap、Spring、MyBatis）的使用，最后在本单元中讲述了购买云直播所需要的所有服务的流程，带领读者熟悉了各个服务的业务流程。

版权所有，严禁传播，违者自负法律责任！

单元 3 搭建开发环境

学习目标

PPT3:
搭建开发环境

知识目标

- 了解 RTMP SDK 的使用方法及原理。
- 了解 JDK 的相关知识。

技能目标

- 掌握 RTMP SDK 的下载与使用方法。
- 掌握 JDK 与 Android Studio 的安装方法。
- 掌握 Android Studio 模拟器的配置方法。

版权所有，严禁传播，违者自负法律责任！

学习情境

公司现在已经将腾讯云服务申请完成，下一步就是开发工作了。公司将此次任务交给了工程师小赵负责。小赵决定使用腾讯云直播提供的 RTMP SDK、Eclipse 和 Android Studio 来作为开发环境。

任务 3.1　了解 RTMP SDK 并下载

任务描述

了解 RTMP SDK 的 API，熟悉其业务流程及内部原理，从腾讯云官网下载移动端直播 SDK，为开发做好准备。

微课 3-1
了解 RTMP SDK 并下载

知识学习

1. RTMP SDK

实时消息传输协议软件开发工具包 SDK（Real Time Messaging Protocol Software Development Kit，SDK）是腾讯云移动终端视频解决方案中的重要组成部分，支持 RTMP 推流、在线直播和点播回看 3 个功能，这 3 个功能的使用方式都遵循一个原则：一个 URL（推流 URL、直播 URL 或者点播 URL）就能驱动。

SDK 囊括了腾讯音/视频团队多年的技术积累，在视频压缩、硬件加速、美颜滤镜、音频降噪、码率控制等方面都做了很多的优化处理。

SDK 由两部分构成：推流器+播放器，本任务将会从这两个方面对推流 SDK 和播放 SDK 进行介绍。

SDK API 涉及的模块如图 3-1 所示。

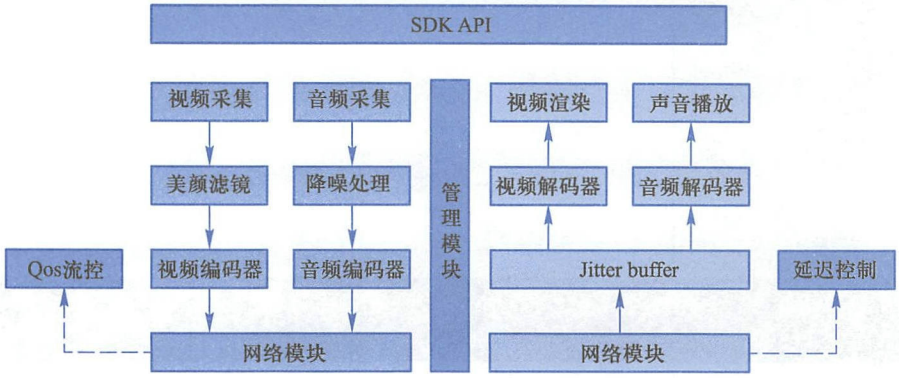


图 3-1
SDK API

2. SDK 对接原理

SDK 业务流程如图 3-2 所示。

版权所有，严禁传播，违者自负法律责任！

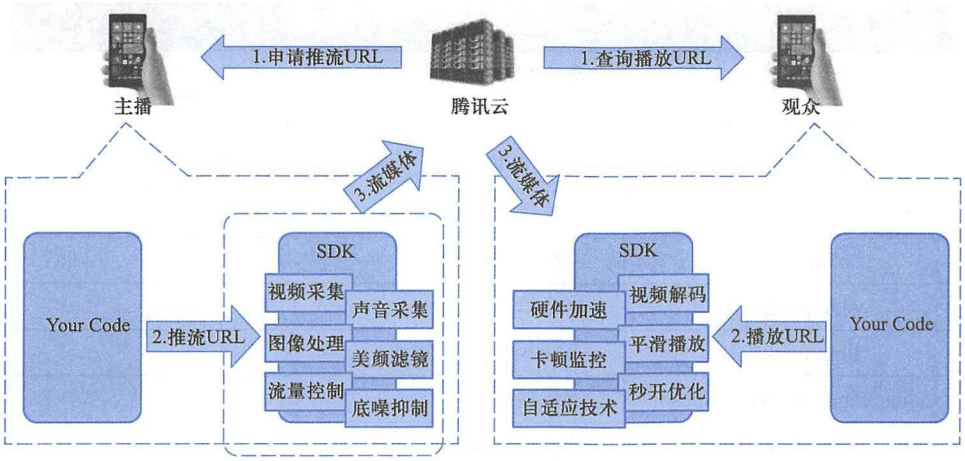


图 3-2
业务流程

腾讯云 RTMP SDK 内部状态机原理如图 3-3 所示。

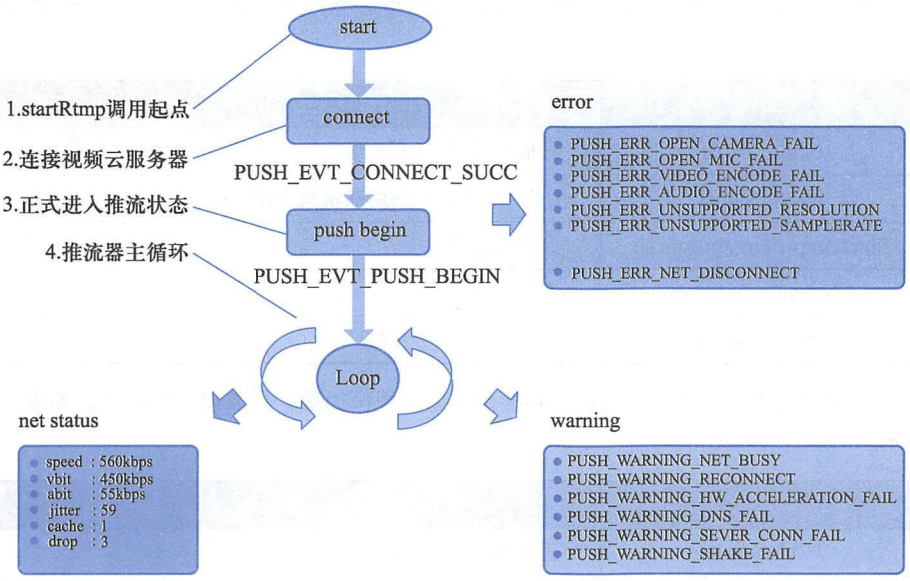


图 3-3
RTMP SDK 内部原理

① 调用的起点：TXLivePush 对象的 startPush 被调用后，推流的流程便会开始启动。

② 连接服务器：推流的第一步是尝试连接推流服务器，如果失败会抛出 NET_DISCONNECT 错误，而不会继续执行后续流程。

③ 开始推流：PUSH_EVT_PUSH_BEGIN 是真正推流开始的标志，很多客户误认为调用了 startPush 就成功开始推流了，其实不然。

④ 推流主循环：直播推流是一个持续的过程，受 SDK 内部一个主循环引擎的驱动，只有在用 stopPush 主动停止或者遭遇不可恢复的错误时才会跳出主循环。

任务实现

首先在腾讯云官网下载 SDK，下载地址：https://www.qcloud.com/document/product/454/7873。下载完成后解压得到 libs 目录，里面主要包含.so 文件和.jar 文件，文件清单如表 3-1 所示。

版权所有，严禁传播，违者自负法律责任！

表 3-1
直播 SDK

文件	说明
txrtmpsdk.jar	直播 SDK Java 层封装
ugcupload.jar	UGC 上传文件到点播系统 JAR 包
sha1utils.jar	UGC 计算上传文件的 SHA 值 JAR 包
cos-sdk-android.1.4.3.6.jar	对象存储 COS 相关的 JAR 包
okio-1.6.0.jar	网络操作 I/O 库
okhttp-3.2.0.jar	网络请求库
libtxrtmpsdk.so	直播 SDK 核心组件
libtraeimp-rtmp-armeabi.so	连麦功能库
libstlport_shared.so	连麦功能库
libTXSHA1.so	UGC 计算上传文件的 SHA 值

如果暂时不需要短视频功能，可以下载完整版，导入以下包即可，如表 3-2 所示。

表 3-2
剔除短视频功能

文件	说明
txrtmpsdk.jar	直播 SDK Java 层封装
ugcupload.jar	UGC 上传文件到点播系统 JAR 包
libtraeimp-rtmp-armeabi.so	连麦功能库
libstlport_shared.so	连麦功能库
libtxrtmpsdk.so	直播 SDK 核心组件

如果暂时不需要连麦及短视频功能，可以下载精简版，导入以下包即可，如表 3-3 所示。

表 3-3
精简版

文件	说明
txrtmpsdk.jar	直播 SDK Java 层封装
ugcupload.jar	UGC 上传文件到点播系统 JAR 包
libtxrtmpsdk.so	直播 SDK 核心组件

注意

SDK 也是需要开发环境的，不需要与 APP 开发环境一致，但要保证兼容。以下是 SDK 的开发环境：

- Android 4.0（API 14）及以上系统。
- Android NDK: android-ndk-r10e。
- Android SDK Tools: android-sdk_r21.1.2。
- minSdkVersion: 14。
- targetSdkVersion: 21。
- Android Studio（推荐使用 Android Studio，也可以使用 Eclipse+ADT）。

项目实训

实训题目

1. 登录腾讯云官网查看 RTMP SDK 的产品文档。
2. 在腾讯云官网查找相关资料来了解 RTMP SDK。

实训目的

了解 RTMP SDK 的特点和可以实现的功能。

实训内容

登录腾讯云官网查看 RTMP SDK 的产品文档。

任务 3.2　安装并配置开发环境

笔 记

任务描述

研究 JDK、Android Studio、Android SDK、Tomcat、Eclipse 的作用及其由来，安装并配置开发环境。

知识学习

1. JDK

JDK（Java Development Kit）称为 Java 开发包或 Java 开发工具，是一个编写 Java 的 Applet 和应用程序的程序开发环境。JDK 是整个 Java 的核心，包括 Java 运行环境（Java Runtime Environment，JRE）、一些 Java 工具和 Java 的核心类库（Java API）。不论什么 Java 应用服务器，实质都是内置了某个版本的 JDK。主流的 JDK 是 Sun 公司（目前已被 Oracle 公司收购）发布的。除了 Sun 之外，还有很多公司和组织都开发了自己的 JDK，例如，IBM 公司开发的 JDK、BEA 公司的 Jrocket，还有 GNU 组织开发的 JDK。

另外，可以把 Java API 类库中的 Java SE API 子集和 Java 虚拟机这两部分统称为 JRE，JRE 是支持 Java 程序运行的标准环境。

JRE 是个运行环境，JDK 是个开发环境。因此写 Java 程序的时候需要 JDK，而运行 Java 程序的时候就需要 JRE。而 JDK 里面已经包含了 JRE，因此只要安装了 JDK，就可以编辑并运行 Java 程序。但由于 JDK 包含了许多与运行无关的内容，占用的空间较大，因此运行普通的 Java 程序无须安装 JDK，而只需要安装 JRE 即可。

JDK 的安装包可以从各个软件资源网站免费获取。2009 年，甲骨文公司宣布收购 Sun。2014 年，甲骨文公司发布了 Java 8 正式版。Java 8 目前是最新版本。

2. Android Studio

Android Studio 是谷歌推出的新的 Android 开发环境，这也是为了方便开发者基于 Android 开发。这个工具与之前用户众多的 Eclipse ADT 相比更加方便、高效。

首先解决的一个问题是多分辨率。Android 设备拥有大量不同规格的屏幕和分辨率，

笔记

开发者可以利用 Android Studio 很方便地调整在各个分辨率设备上的应用。

其次，Android Studio 还解决了语言问题。拥有多语言版本（但是没有中文版本）、支持翻译的功能都让开发者更适应全球开发环境。Android Studio 还提供收入记录功能。

最大的改变在于 Beta 测试的功能。Android Studio 提供了 Beta Testing，可以让开发者很方便地试运行。

另外，该工具的代码提示和搜索功能非常强大、非常智能。例如，开发者自定义 theme 有个名字，叫作 light_play_card_bg.xml，如果在 Eclipse 里，用户必须要输入 light 开头才能提示下面的内容，而在 Android Studio 里，用户只需要输入其中的任意一段，如 card，下面就会出现提示的内容。

Android Studio 会智能预测并给开发者最优的提示。每一次并非都是相同的提示结果，而可能是用户最想用、最可能用的结果。

Android Studio 相对 Eclipse 比较明显的优势有如下几点。

① 颜色、图片在布局和代码中可以实时预览。

② 多屏预览、截图带有设备框，可随时录制模拟器视频。

③ 可以直接打开文件所在位置。

④ 跨工程移动、搜索和跳转。

⑤ 自动保存，无须一直按 Ctrl + S 组合键。

⑥ 即使文件关闭，依然可以回退多个历史。

⑦ 智能重构和智能预测报错。

⑧ 每一行文件编辑历史可追溯到人。

⑨ 自带较强大的图片编辑功能等。

正是因为 Android Studio 具有较多方便开发者的地方，目前使用者越来越多。

3. Android SDK

软件开发工具包（Software Development Kit，SDK）被定义为特定的软件包、软件框架、硬件平台、操作系统等建立应用软件的开发工具集合。Android SDK 即开发 Android 应用程序的工具包。

Android SDK 采用了 Java 语言，在安装 Android Studio 的时候设定好路径，系统会自动下载安装。

Android SDK 目录下有很多文件夹，下面分别介绍各个文件夹的作用。

① add-ons 文件夹用来存放附加库，即第三方公司为 Android 平台开发的附加功能系统，如 GoogleMaps，如果用户安装了 OphoneSDK，也会有一些类库在里面。

② docs 文件夹用来存放 Android SDK API 参考文档，所有的 API 都可以在这里查到。

③ market_licensing 文件夹作为 AndroidMarket 版权保护组件，一般发布付费应用到电子市场可以用它来反盗版。

④ platforms 文件夹是每个平台 SDK 真正的文件，里面会根据 API Level 划分的 SDK 版本。这里就以 Android 2.2 为例，进入后有一个名为 android-8 的文件夹，进入后是 Android 2.2 SDK 的主要文件。其中 ant 目录存放 ant 编译脚本，data 目录存放一些系统资源，images 目录存放模拟器映像文件，skins 目录存放 Android 模拟器的皮肤，templates 目录存放工程创建的默认模板，android.jar 目录存放该版本的主要 framework 文件，tools 目录里面包含

了重要的编译工具，如 aapt、aidl、逆向调试工具 dexdump 和编译脚本 dx。

⑤ platform-tools 文件夹保存着一些通用工具，如 adb 和 aapt、aidl、dx 等文件。Android123 提示，这里和 platforms 目录中 tools 文件夹有些重复，主要是因为从 Android 2.3 开始这些工具被划分为通用了。

⑥ samples 文件夹是 Android SDK 自带的默认示例工程，里面的 API Demos 强烈推荐初学者运行学习。对于 SQLite 数据库操作可以查看 NotePad 这个例子，对于游戏开发，Snake、LunarLander 都是不错的例子；对于 Android 主题开发，Home 则是 Android 5 时代的主题设计原理。

⑦ tools 文件夹包含许多重要的工具，如用于启动 Android 调试工具 Android 开发环境中的 Dalvik 虚拟机调试监控服务（Dalvik Debug Monitor Service，DDMS）、logcat、屏幕截图和文件管理器等。draw9patch 则是绘制 Android 平台的可缩放 PNG 图片的工具；sqlite3 是可以在 PC 上操作 SQLite 数据库的；monkeyrunner 则是一个不错的压力测试应用，它可以模拟用户随机按键；mksdcard 是模拟器 SD 映像的创建工具；emulator 是 Android SDK 模拟器主程序，不过从 Android 1.5 开始，需要输入合适的参数才能启动模拟器；traceview 则作为 Android 平台上重要的调试工具。

⑧ build-tools 文件夹里有各个版本模拟器在 Android 平台的相关通用工具，如 AAPT、aidl、dx 等文件。AAPT 即 Android Asset Packaging Tool，在 SDK 的 build-tools 目录下，该工具可以查看、创建和更新 ZIP 格式的文档附件（如.zip、.jar、.apk），也可将资源文件编译成二进制文件。

⑨ extras 文件夹下存放了谷歌提供的 USB 驱动、英特尔提供的硬件加速等附加工具包。

4. Tomcat

Tomcat 是一个免费并且开源的 Web 服务器，它是 Apache 软件基金会的 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其他一些公司及个人共同开发而成。由于有了 Sun 公司的参与和支持，最新的 Servlet 和 JSP 规范总是能在 Tomcat 中得到体现。Tomcat 因其技术先进、性能稳定、简单易用，成为目前应用最为广泛的 Web 服务器。

5. Eclipse

Eclipse 是一种可扩展的开放源代码的集成开发环境，是一款非常受欢迎的 Java 开发工具。Eclipse 的最大特点是接受各种开放源代码的插件，其中就包括 Java 开发工具（Java Development Tools，JDT），利用它可以大大提高开发效率。

任务实现

微课 3-2
安装并配置开发环境

1. 开发环境安装

（1）安装配置 JDK

① 获取 JDK 安装程序。

可在 Oracle 官网地址下载需要的 JavaSE 安装文件，地址如下：http://www.oracle.com/technetwork/java/javase/downloads/index.html。

② 打开 JDK 安装程序，弹出对话框，如图 3-4 所示。

图 3-4
开始安装 Java SDK

③ 设置安装路径，如图 3-5 所示。

图 3-5
设置安装路径

单击“更改”按钮，可以在弹出的窗口中修改安装路径。单击“下一步”按钮，安装 JDK。在这个步骤中可以在可选安装列表框中选择要安装的内容。

④ 安装 JRE。

安装完 JDK 后会弹出 JRE 安装窗口（此处需要注意，修改安装路径必须和 JDK 在同一目录下，而不是安装在 JDK 目录下），单击“下一步”按钮进行安装，如图 3-6 所示。

图 3-6
JRE 安装窗口

⑤ 验证安装是否成功。

安装完成后，需要验证 Java 环境是否安装成功。验证方式有多种，可以编写一个最简单的 Java 程序文件，然后编译执行它，也可以通过在 CMD 窗口输入 java -version 命令进行验证，如图 3-7 所示。

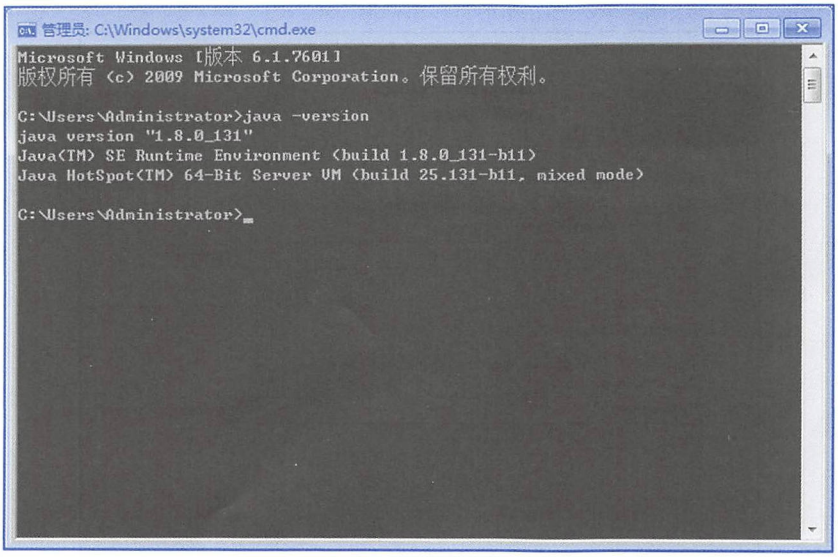


图 3-7
验证 Java 环境安装情况

(2) 安装配置 Android Studio

① 获取 Android Studio。

进入官网 (https://developer.android.com/studio/index.html)，下载包含 Android SDK 的 Android Studio。本任务采用的是 Android Studio 2.1.1 版本。

下载完成后，打开安装向导。

② 选择需要安装的组件。

读者可以使用默认选项，也可以根据自己的需求在列表框内进行选择，如图 3-8 所示。

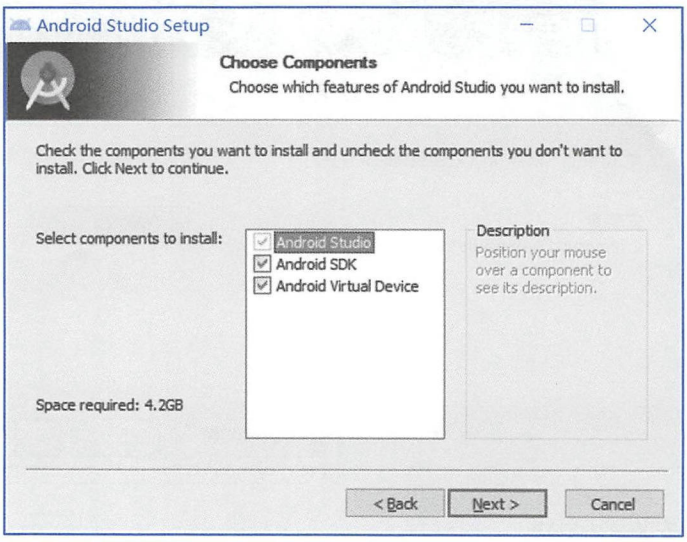


图 3-8
选择需要安装的组件

③ 选择安装目录。

选择 Android Studio 的安装目录和 SDK 的安装目录。然后单击“Next”按钮，如图 3-9 所示。

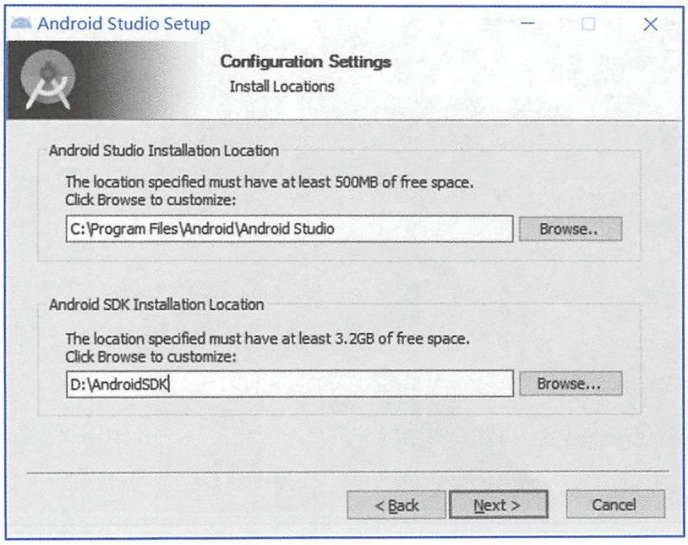


图 3-9
选择安装目录

由于 SDK 需要单独下载，并且后面在使用过程中会下载配置 Android 模拟器，需要大量的存储空间，因此建议安装 SDK 的文件夹在 Windows 环境中单独选择磁盘空间比较大的分区。

④ 导入配置选项。

待安装完成后，会弹出对话框，提示是否导入 Android Studio 以前的一些配置。如果是全新安装这个软件或者不需要导入之前版本的环境设置，这里就可以选择第二项，然后单击“OK”按钮，具体如图 3-10 所示。

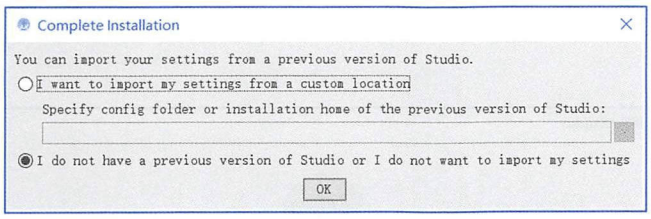


图 3-10
选择是否导入之前版本的环境设置

⑤ 选择 UI 主题。

接下来选择 Android Studio 的 UI 主题，这个可以根据自己的偏好选择。选择完成后单击“Next”按钮，如图 3-11 所示。

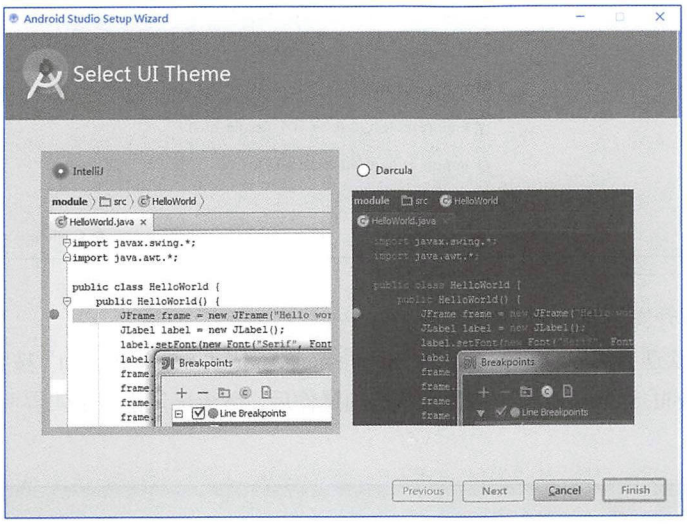


图 3-11
选择 Android Studio 的 UI 主题

⑥ 组件下载。

安装完毕后，打开安装好的 Android Studio。第一次启动需要下载 SDK 等一些组件，如图 3-12 所示。

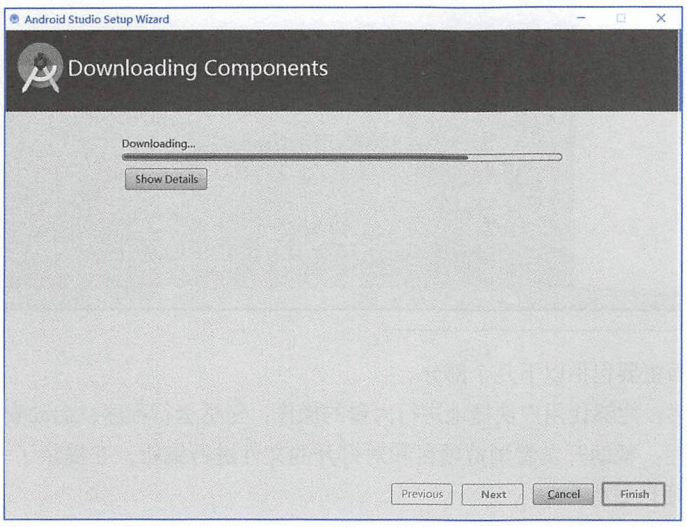


图 3-12
下载组件

待软件打开后，Android Studio 的安装已经全部完成，会出现如图 3-13 所示的启动欢迎界面。

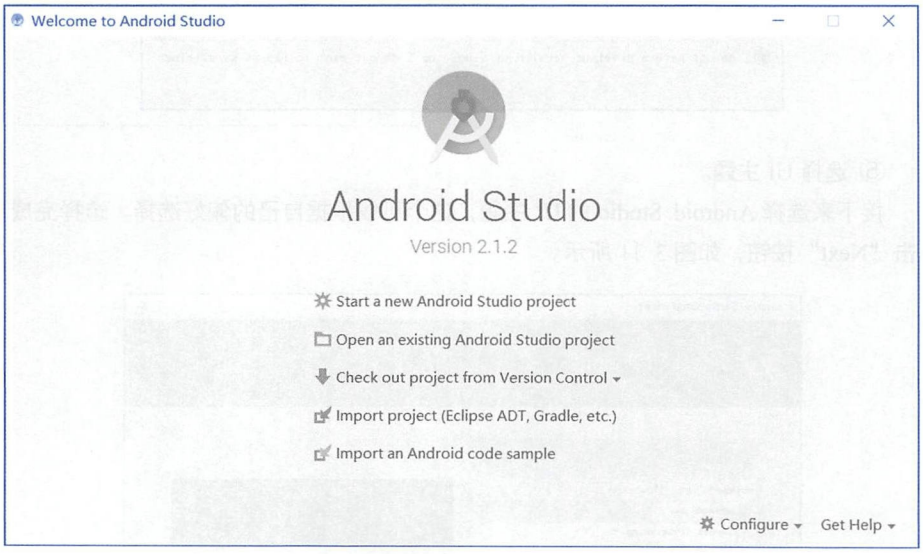


图 3-13
启动欢迎界面

⑦ 了解开发界面。

在启动欢迎界面中，单击第一个“Start a new Android Studio Project”按钮可以创建一个新的 Android Studio 工程，按照默认选项操作后，就会出现工程编辑界面，如图 3-14 所示。

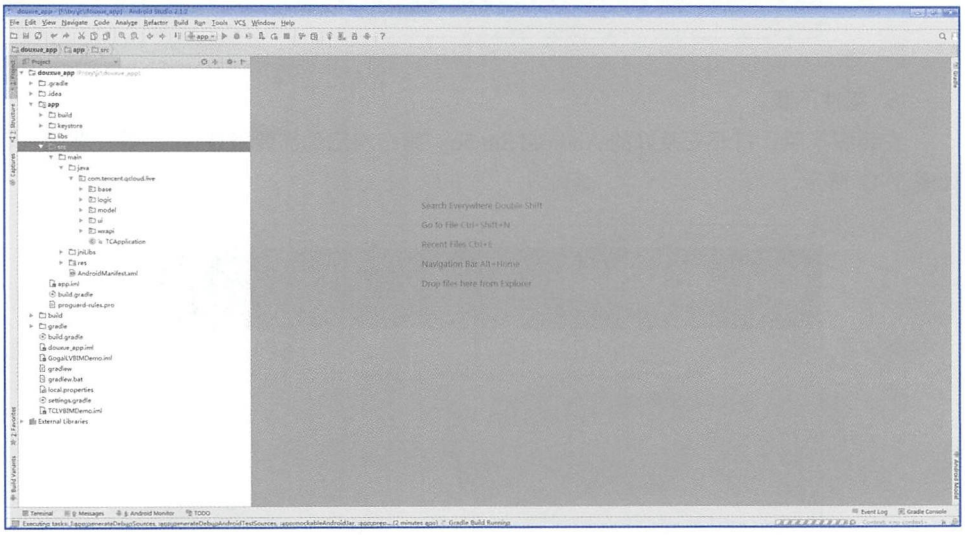


图 3-14
工程编辑界面

整个界面主要包括以下几个部分。

- 工具栏：能够让用户快捷地进行大量的操作，包括运行程序、启动安卓工具等。
- 导航栏：帮助开发者浏览项目和对打开的文件进行编辑，它提供了一个更紧凑可见的项目结构。
- 编辑窗口：是开发者编辑和修改代码的地方，这个窗口会根据文件类型的改变而改变。

- 工具窗口：让开发者可以访问指定的任务，如项目管理、搜索、版本控制等。
- 状态栏：展示了项目的状态，以及一些警告或者消息。

（3）安装配置 Tomcat

① 下载安装。

从 Apache 网站（http://tomcat.apache.org/）下载 Tomcat，当前最新版本为 9.0，本书采用 Tomcat 7.0 版本开发。

将下载好的 Tomcat 压缩包解压至本地磁盘，最好选择较浅的路径进行安装，如磁盘的根路径，解压好文件，如图 3-15 所示。

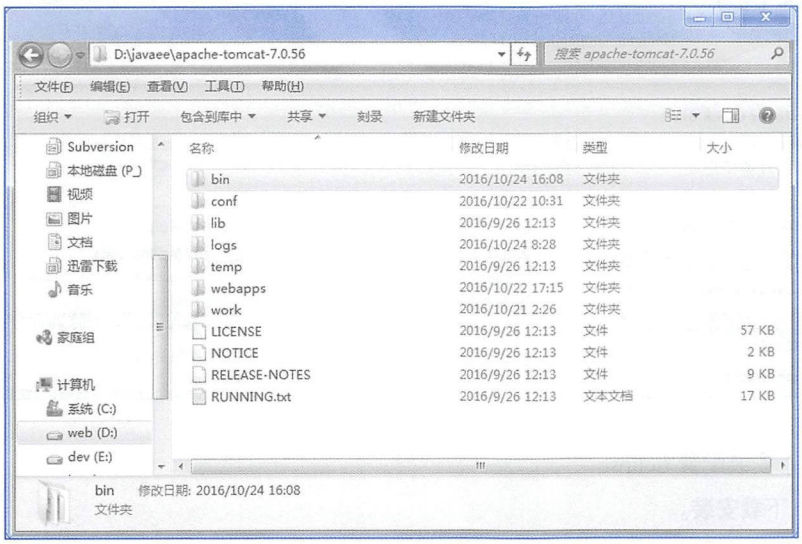


图 3-15
Tomcat 目录结构

② 运行测试。

在 Tomcat 文件夹下进入 bin 文件夹，双击 startup.bat 文件，启动 Tomcat 服务器，启动成功后，如图 3-16 所示。

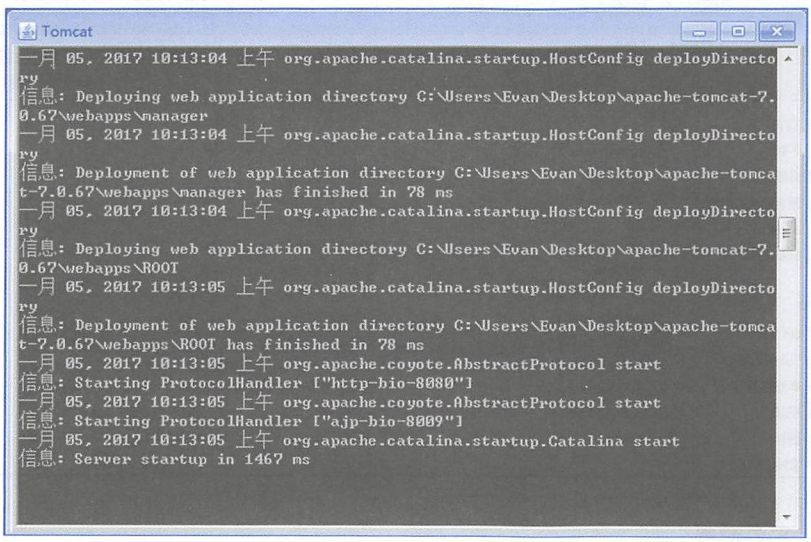


图 3-16
启动 Tomcat 服务器

在浏览器的地址栏内访问 http://localhost:8080，显示如图 3-17 所示的界面，表示 Tomcat 服务器部署成功。

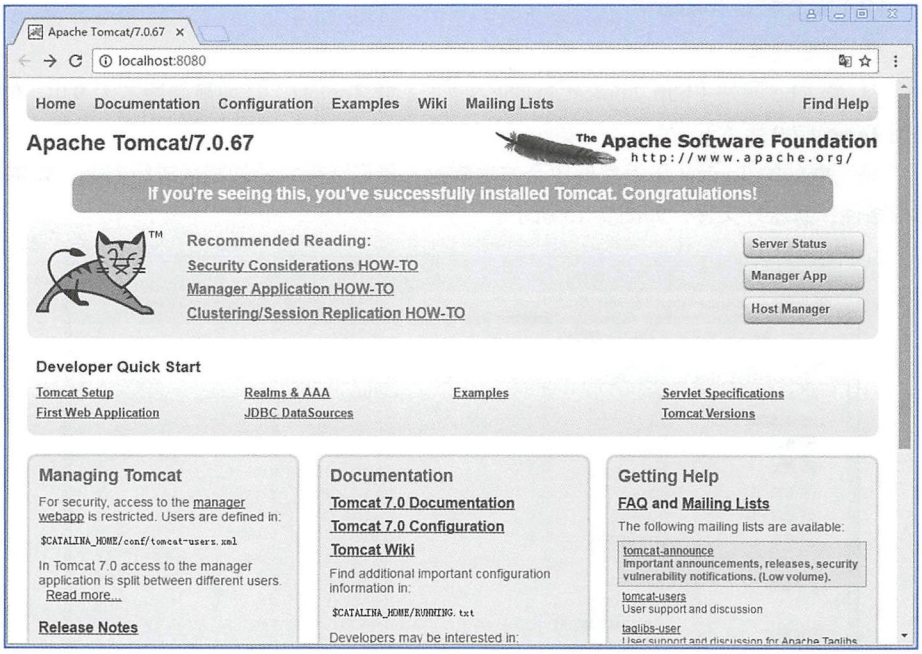


图 3-17
Tomcat 登录界面

(4) 安装配置 Eclipse

① 下载安装。

读者可以从www.eclipse.org网站获取到最新版本的 Eclipse，一般选择下载 Eclipse IDE for Java EE Developers 版本。

将下载好的 Eclipse 压缩包解压至计算机本地磁盘，解压好的文件如图 3-18 所示。

图 3-18
Eclipse 目录结构

② 运行配置。

双击 eclipse.exe 文件启动 Eclipse，选择工作目录，该目录不要有中文路径，最后单

击“OK”按钮，如图 3-19 所示。

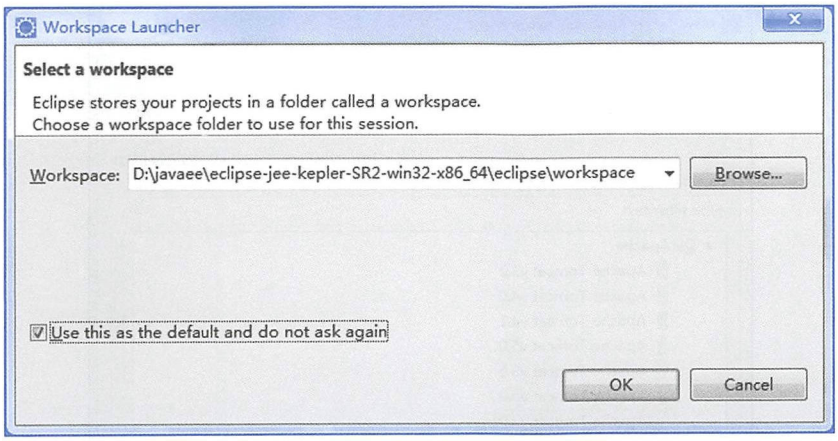


图 3-19
Eclipse 主界面

③ 配置 Eclipse 的 Web 服务器。

在主菜单栏中选择“Window→Preferences→Server→Runtime Environments”菜单命令，进入如图 3-20 所示的属性界面。单击右侧的“Add...”按钮添加 Tomcat 服务器。

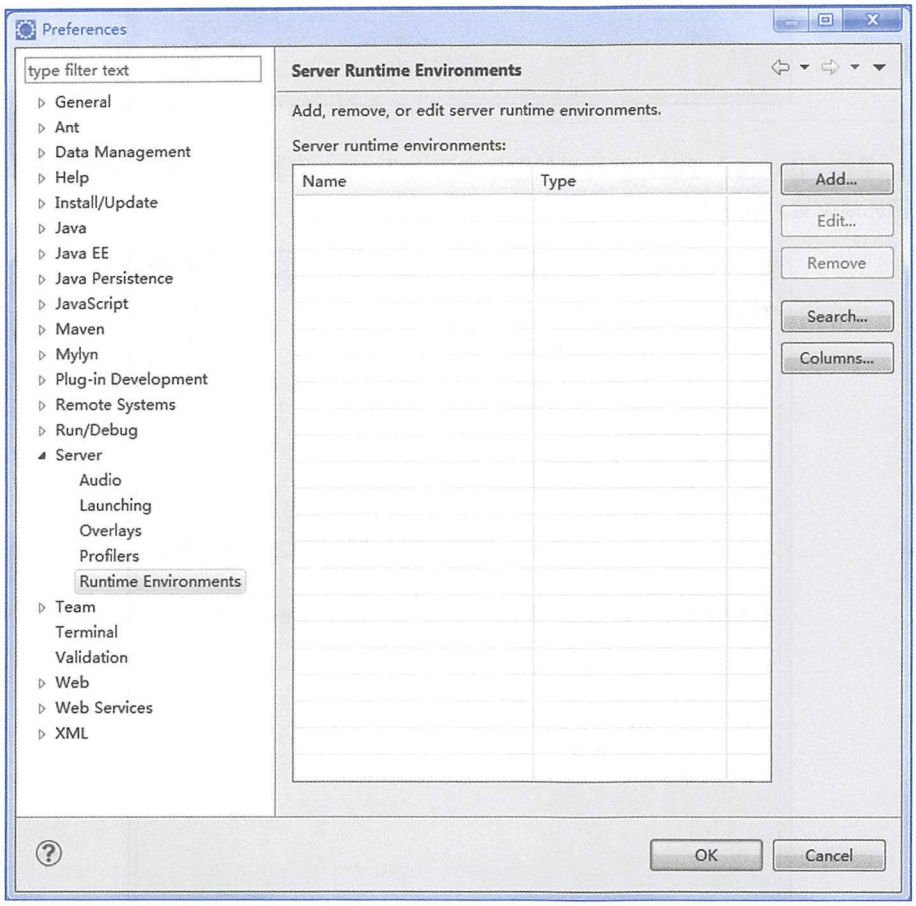


图 3-20
服务运行环境

选择 Apache 文件夹下的 Apache Tomcat v7.0，单击“Next”按钮，如图 3-21 所示。

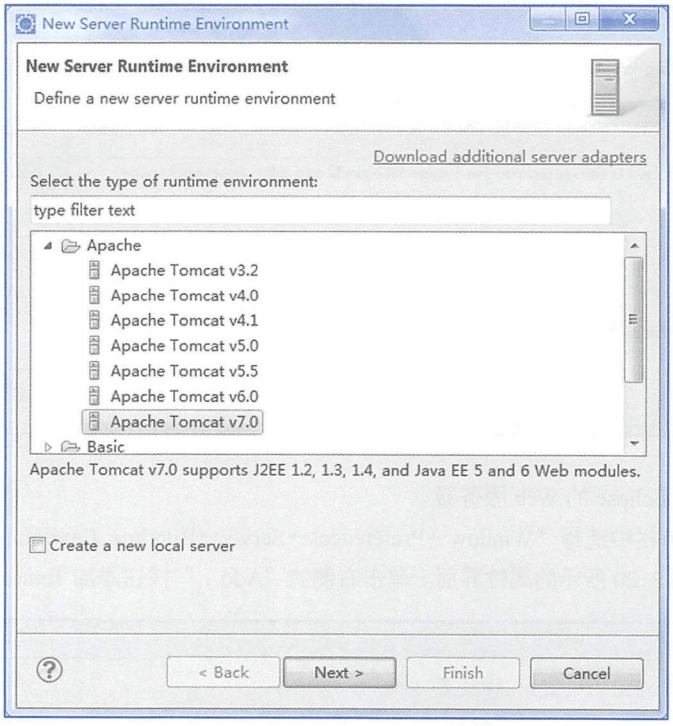


图 3-21
选择 Tomcat 服务器

单击“Browse”按钮，在弹出的窗口中选择 Tomcat 安装路径，最后单击“Finish”按钮完成配置，效果如图 3-22 所示。

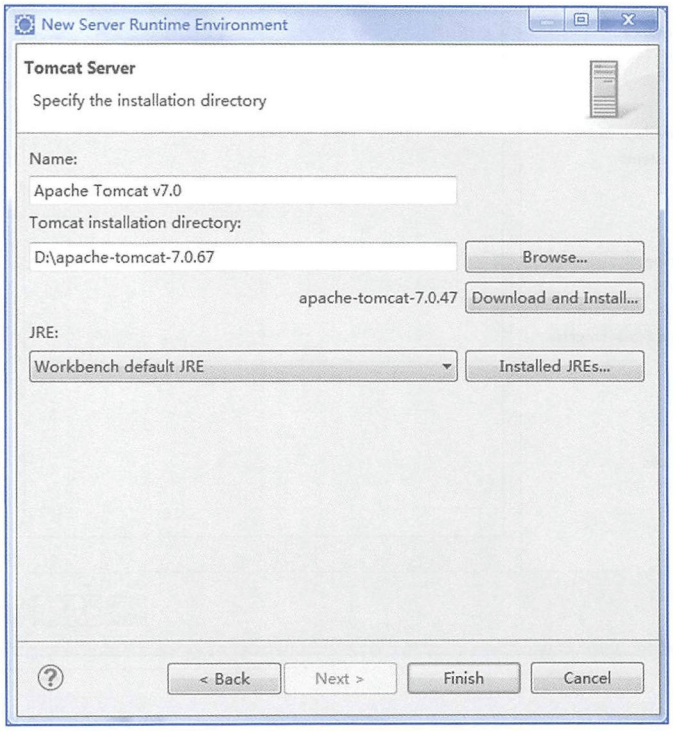


图 3-22
配置 Tomcat

在主菜单栏中选择“Window→Show View→Servers”菜单命令，打开 Servers 窗口，添加刚刚创建的 Tomcat 服务器，在此窗口中可以直接停止或者启动调试 Tomcat 服务器，如图 3-23 所示。

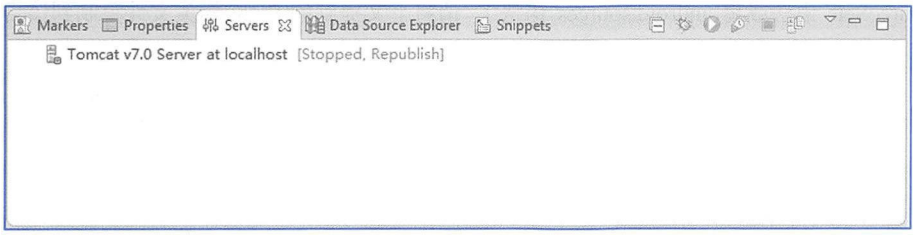


图 3-23
添加 Tomcat 服务器

在主菜单栏中选择“Window→Preferences→General→Workspace”命令，在 Text file encoding 下设置工作空间的代码编码方式为 UTF-8，完成后单击“OK”按钮，效果如图 3-24 所示。

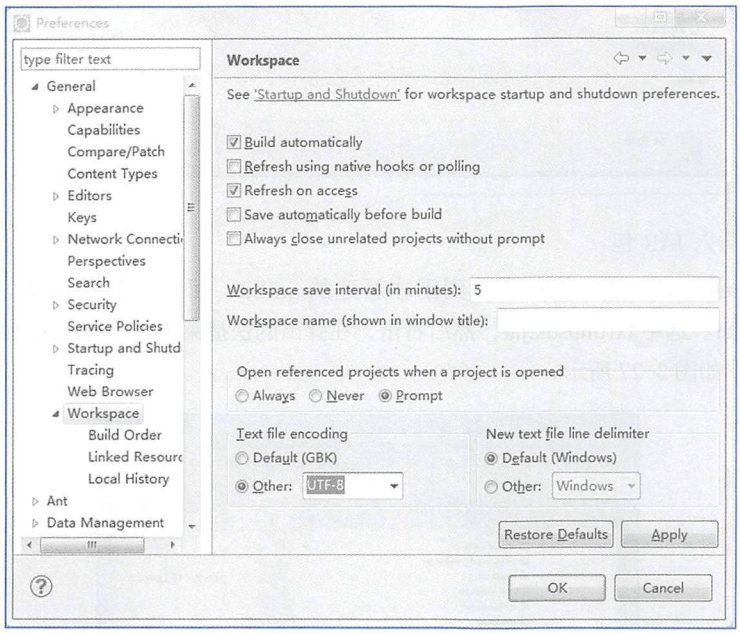


图 3-24
设置编码

2. Android Studio 配置

(1) 新建工程

新建 Android 工程，如图 3-25 所示。

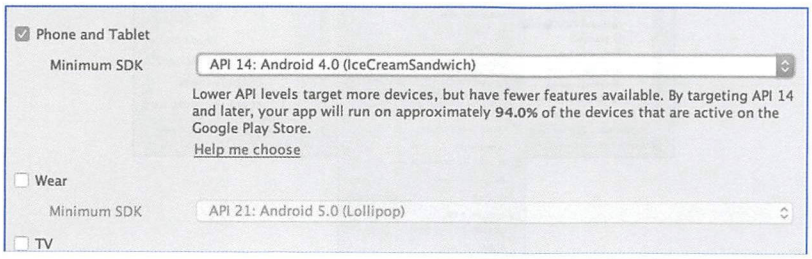


图 3-25
新建 Android 工程

（2）复制 SDK 文件

如果之前没有为工程指定过 jni 的加载路径，推荐放在/src/main/jniLibs 目录下，这是 Android Studio 默认的 jni 加载目录。如果已经指定过 jni 的加载路径（通过 gradle 的语法：sourceSets 语法或者 android.sources 语法），请将上文提到的 SDK 相关文件复制到该目录下，如图 3-26 所示。

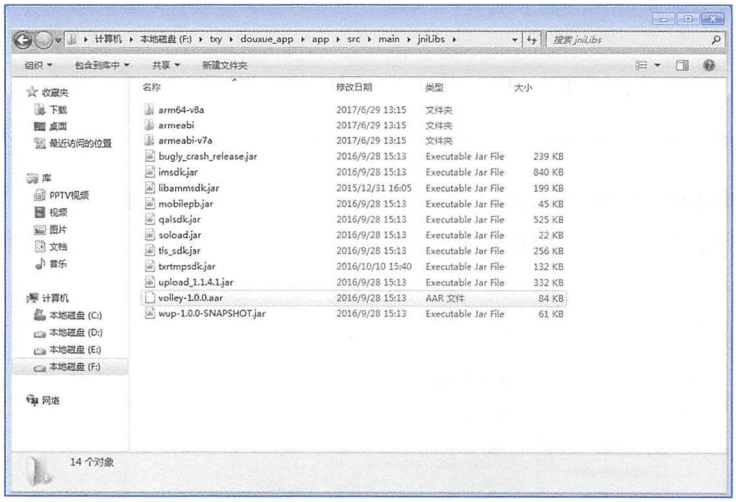


图 3-26
复制文件

（3）导入 JAR 包

在 Android Studio 工程中找到刚才新建的 jniLibs 目录，展开目录，可以看到 txrtmpsdk.jar。选中 txrtmpsdk.jar，然后右击，在弹出的快捷菜单中选择“Add As Library”菜单命令，如图 3-27 所示。

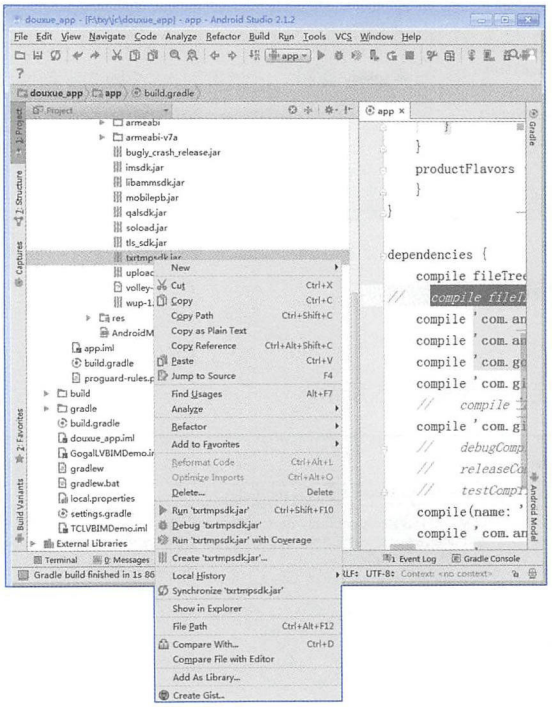


图 3-27
导入 JAR 包

导入之后，可以看到 build.gradle 中自动生成以下这一行脚本，如图 3-28 所示。

```
compile(name: 'volley-1.0.0', ext: 'aar')
compile 'com.android.support:multidex:1.0.1'
compile 'com.android.support:support-v4:24.2.1'
compile files('src/main/jniLibs/txrtmpsdk.jar')
```

图 3-28 添加依赖

(4) 配置 APP 权限

在 AndroidManifest.xml 中配置 APP 的权限，音/视频类 APP 一般需要以下权限。

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_LOGS"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-feature android:name="android.hardware.Camera"/>
<uses-feature android:name="android.hardware.camera.autofocus"/>
```

(5) 验证配置

在工程中调用 SDK 接口，获取 SDK 版本信息，以验证工程设置是否正确。

① 引用 SDK。

在 MainActivity.java 中引用 SDK 的 class。

```
import com.tencent.rtmp.TXLivePusher;
```

在 onCreate 中调用 getSDKVersion 接口获取版本号。

```
int[] sdkver = TXLivePusher.getSDKVersion();
if (sdkver != null && sdkver.length >= 3) {
    Log.d("rtmpsdk","rtmp sdk version is:" + sdkver[0] + "." + sdkver[1] + "." + sdkver[2]);
}
```

② 编译运行。

如果前面各步骤都操作正确，Demo 工程将顺利编译通过，运行之后将在 logcat 中看到如下 log 信息。

```
07-13 20:25:05.099 26119-26119/? D/rtmpsdk: rtmp sdk version is:1.5.188
```

③ 问题排查。

如果将 SDK 导入到工程以后，编译运行出现类似以下错误。

```
Caused by: android.view.InflateException:
Binary XML file #14:Error inflating class com.tencent.rtmp.ui.TXCloudVideoView
```

可以按照以下流程来排查问题。

- 确认是否已经将 SDK 中的 JAR 包和 so 库放在 jniLabs 目录下。
- 如果使用完整版本，确认是否将 x64 架构的 so 库过滤掉。因为完整版本中的连麦功能暂时不支持 x64 架构的手机。

```
buildTypes {
    release {
        minifyEnabled  false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        ndk {
            // 需要将 armeabi-x64 架构过滤掉
            abiFilters "armeabi", "armeabi-v7a"
        }
    }
}
```

- 检查下混淆规则，确认是否将 SDK 相关的类也给混淆了。

（6）Android Studio 模拟器配置

单击快捷工具栏中的模拟器按钮，出现模拟器选择页面，如图 3-29 所示。

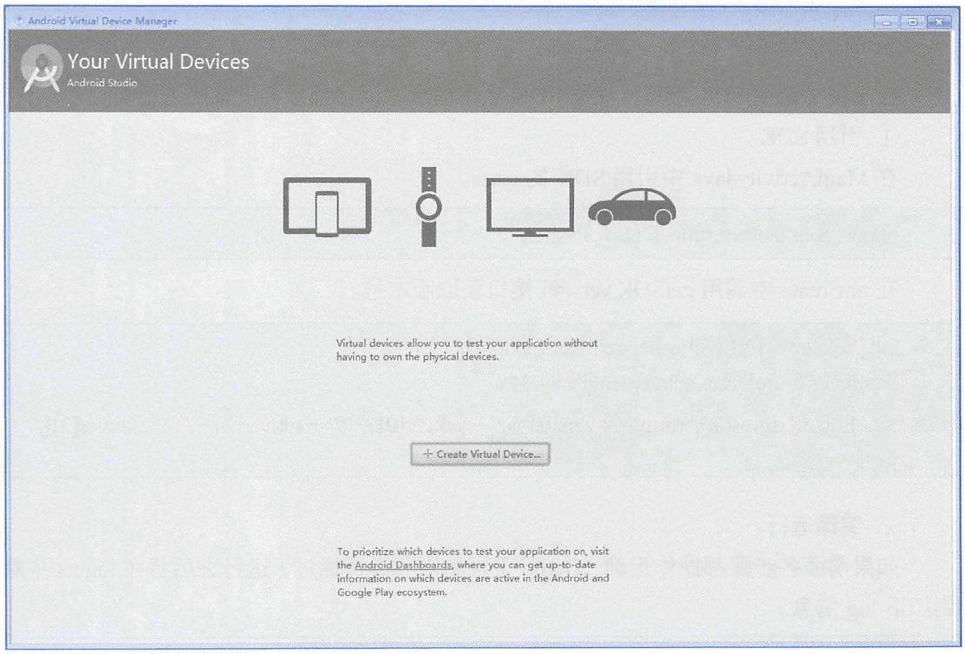


图 3-29
模拟器选择界面

单击“Create Virtual Device”按钮，弹出模拟器类型选择界面，如图 3-30 所示。选择“Phone→4.7"WXGA”选项，单击“Next”按钮。

跳转到模拟器图像设置界面，单击“Other Images”选项卡，选中一个“ABI”为“arm64-v8a”的选项，单击“Next”按钮，如图 3-31 所示。

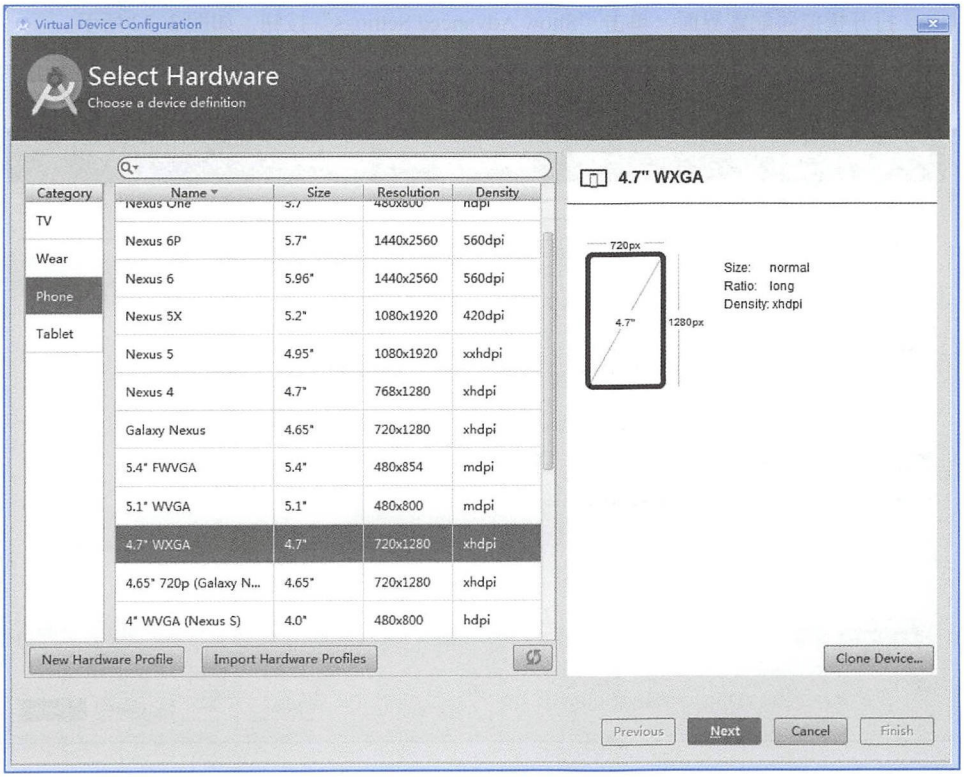


图 3-30
类型选择界面

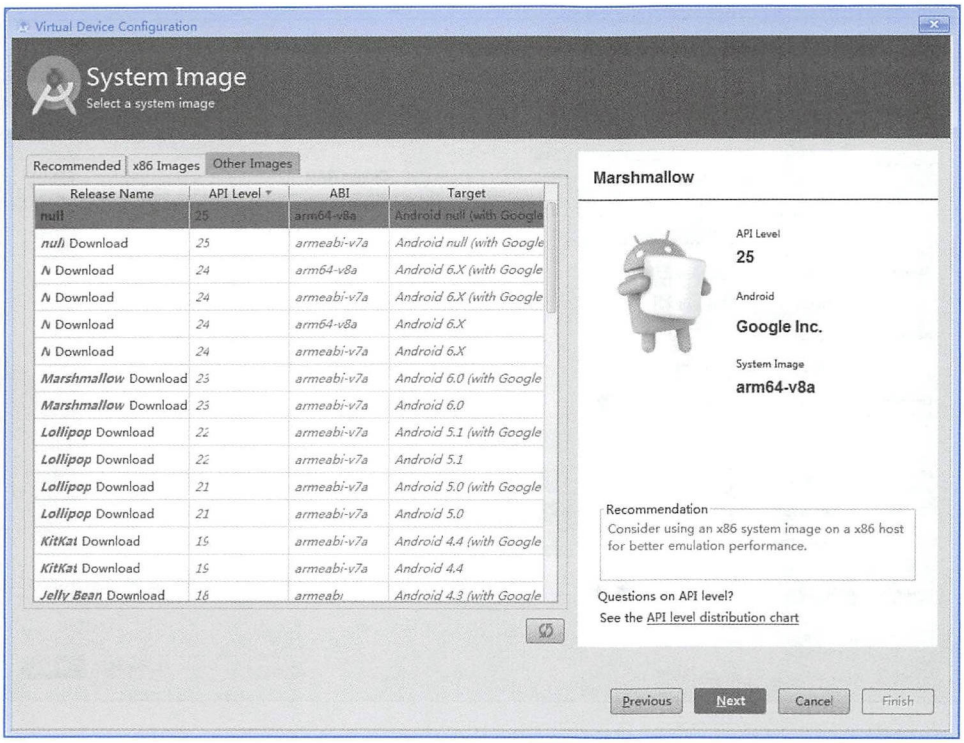


图 3-31
图像设置界面

打开模拟器配置界面，单击“Show Advanced Settings”按钮，如图 3-32 所示。

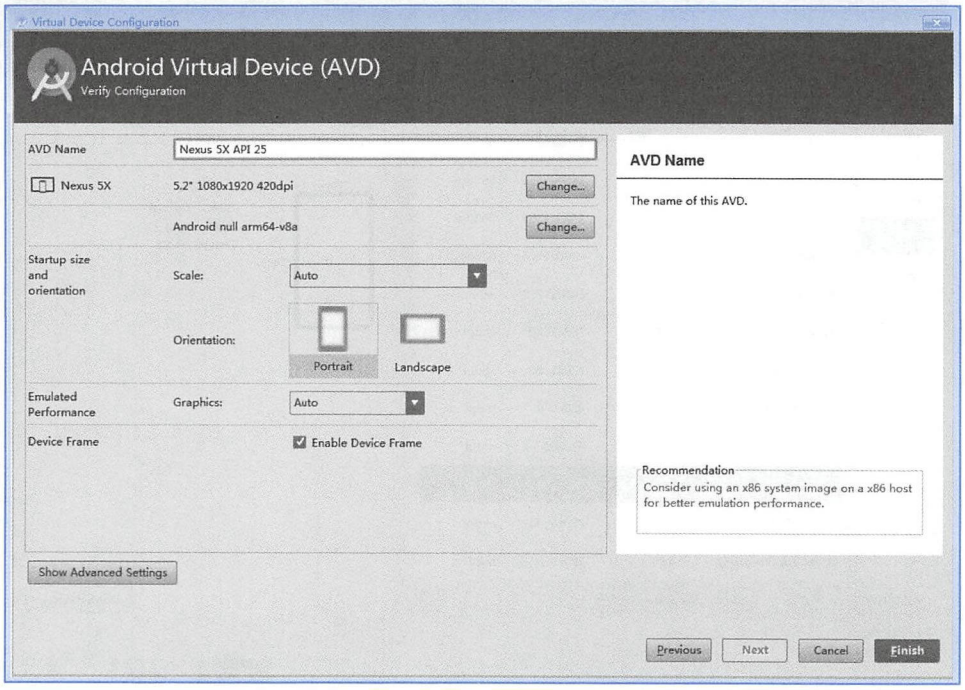


图 3-32
模拟器配置界面

将配置设置为如图 3-33 所示。

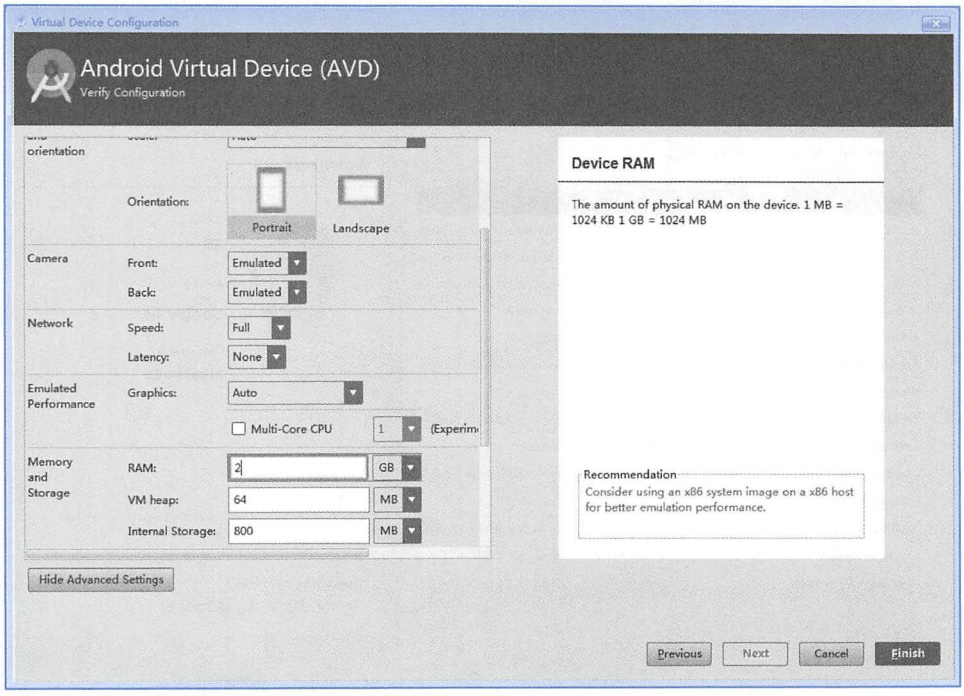


图 3-33
配置完成界面

配置完成后单击“Finish”按钮，打开设备管理器窗口，如图 3-34 所示。

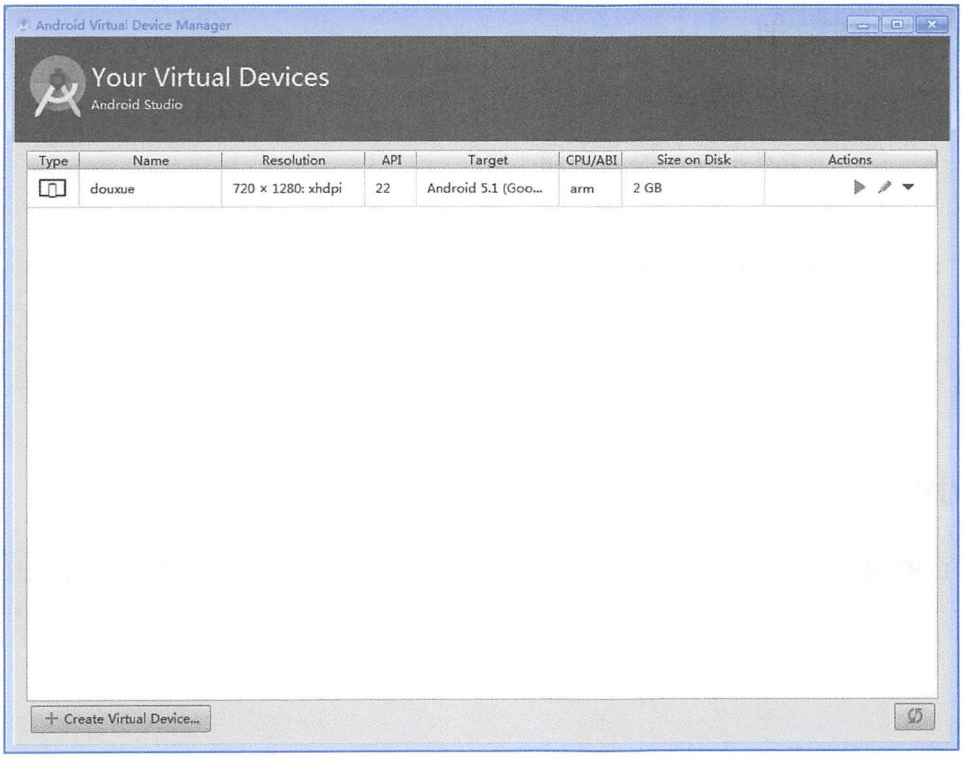


图 3-34
模拟器启动选择界面

单击所选设备“Actions”下的“开始”按钮▶，启动模拟器，出现模拟器界面，等待启动完成，如图 3-35 所示。

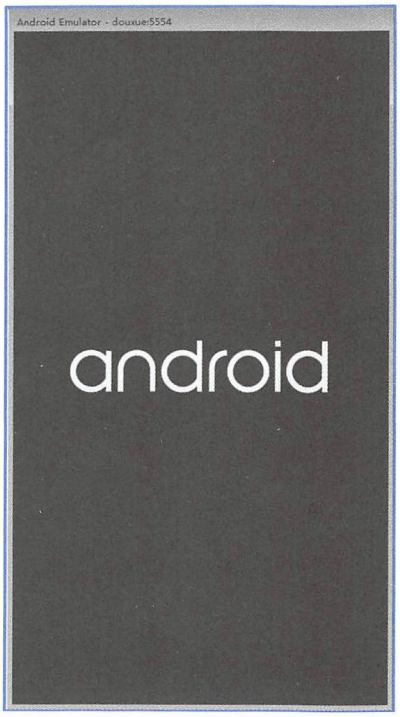


图 3-35
模拟器启动中

项目实训

实训题目

完成环境配置并导入一个 Android 工程。

实训目的

熟练掌握 Android Studio 的配置并学会使用 Android Studio。

实训内容

从 GitHub 上获取一个 Android 项目导入配置完成并运行。

单元小结

本单元讲解了 RTMP SDK 的作用及使用方法，重点阐述如何配置开发环境，及验证配置是否完善。

版权所有，严禁传播，违者自负法律责任！

单元 4

用户管理模块

学习目标

PPT4:
用户管理模块

知识目标

- 了解 HTTP 的 Get 和 Post 方式。
- 了解 Activity 跳转。
- 了解 Android Studio 导入 jar 包的方法。

技能目标

- 掌握用户注册功能的实现方法。
- 掌握用户登录功能的实现方法。
- 掌握个人中心的实现方法。

版权所有，严禁传播，违者自负法律责任！

学习情境

公司将开发直播 APP 的任务交给了工程师小赵。小赵已经完成了系统设计和开发环境的配置，接下来的工作就是开发具体的功能模块了。小赵决定先完成用户管理模块，要求直播 APP 能支持用户登录、注册和管理个人信息。

任务 4.1　实现用户注册模块

任务描述

Web 后台实现用户注册功能的接口，提供给 Android 程序调用。判断用户是否存在，如果不存在，则创建新用户，保存至数据库，密码用 MD5 加密。APP 先实现对用户输入的账号密码进行判断，确定其是否符合规范，如果符合规范，则调用后台提供的接口进行注册。

知识学习

Get 与 Post 的主要区别

Get 与 Post 都是基于 HTTP 协议的网络数据交互方式。它们的区别主要在于以下几点。

① Get 通常是从服务器上获取数据的；Post 通常是向服务器传送数据的。

② Get 是把参数数据队列加到表单的 ACTION 属性所指的 URL 中，值和表单内的各个字段一一对应，在 URL 中可以看到，实际上就是 URL 拼接方式；Post 是通过 HTTP Post 机制，将表单内各个字段与其内容放置在 HTML HEADER 内一起传送到 ACTION 属性所指的 URL 地址。

③ 对于 Get 方式，服务器端用 Request.QueryString 获取变量的值；对于 Post 方式，服务器端用 Request.Form 获取提交的数据。

④ Get 传送的数据量较小，不能大于 1 KB；Post 传送的数据量较大，一般被默认为不受限制。但理论上，IIS4 中最大量为 80 KB，IIS5 中最大量为 100 KB。

⑤ Get 安全性非常低；Post 安全性较高。

微课 4-1
实现用户注册模块

任务实现

用户注册功能具体流程如图 4-1 所示。

1. 后台用户注册接口开发

源码 4-1：
实现用户注册模块

① 打开 Eclipse，导入本教材提供的配套项目包，此项目包是一个没有实现具体功能的空工程。

版权所有，严禁传播，违者自负法律责任！

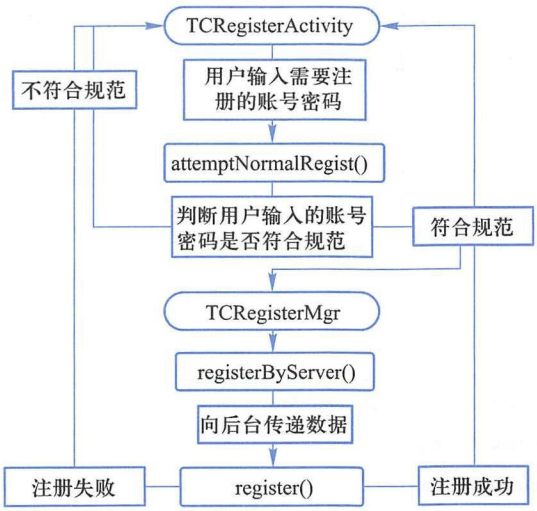


图 4-1
用户注册流程图

② 在 com.douxue.live.controller.app.user 中的 UserController.java 文件中实现注册功能的接口，添加的代码如下。

```
@Autowired
private UserService userService;
/**
 * 用户注册
 * @param account
 * @param password
 * @return
 *
 */
@RequestMapping(value = "register", method = RequestMethod.POST)
@ResponseBody
public RestResult register(@RequestParam("account") String account,
                           @RequestParam("password") String password) {
    return userService.register(account, password);
}
```

③ 在 com.douxue.live.service.user 中的 UserService.java 文件中添加代码，代码如下所示。

```
/**
 * 用户注册
 * @param account
 * @param password
 * @return
 */
RestResult register(String account, String password);
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

④ 这是一个接口类，所以要在 com.douxue.live.service.user.impl 中的 UserServiceImpl.java 文件中来实现接口方法，代码如下。

```
@Autowired
private UserMapper userMapper;

/**
 * 用户注册
 */
@Override
public RestResult register(String account, String password) {
    User user = new User();
    user.setAccount(account);
    User verify = userMapper.findOneUser(user);
    if (null != verify) {
        return new RestResult(ErrorCode.ERROR_ACCOUNT_IS_EXIST);
    }
    user.setUserId(UUIDUtils.getUUID());
    user.setPassword(MD5Utils.encryptUserPassword(password));
    user.setAccountType(EnumUser.ACCOUNT_TYPE_COMMON.getValue());
    // 用户昵称默认为用户账号，可以不设置或者重新定制方案
    user.setNickName(account);
    if(userMapper.insertSelective(user) > 0){
        user = userMapper.findOneUser(user);
        user.setPassword(null);
        user.setWxUnionid(null);
        return new RestResult(user);
    }
    return RestResult.FAIL;
}
```

⑤ 在 com.douxue.live.dao.mapper 中的 UserMapper.java 文件中实现向数据库添加数据的方法接口，代码如下。

```
//插入数据
int insertSelective(User record);
//查询数据
User findOneUser(User user);
```

⑥ 在 src/main/resources/mapper 中的 UserMapper.xml 文件中实现具体方法，代码如下。

```
<insert id="insertSelective" parameterType="com.mz.live.dao.entity.User" useGeneratedKeys=
"true" keyProperty="userId">
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用


```
insert into t_user
    <trim prefix="(" suffix=")" suffixOverrides=",">
        <if test="userId != null">
            user_id,
        </if>
        <if test="wxUnionid != null">
            wx_unionid,
        </if>
        <if test="account != null">
            account,
        </if>
        <if test="password != null">
            password,
        </if>
        <if test="accountType != null">
            account_type,
        </if>
        <if test="phone != null">
            phone,
        </if>
        <if test="headPic != null">
            head_pic,
        </if>
        <if test="userName != null">
            user_name,
        </if>
        <if test="nickName != null">
            nick_name,
        </if>
        <if test="state != null">
            state,
        </if>
        create_time
    </trim>
    <trim prefix="values (" suffix=")" suffixOverrides=",">
        <if test="userId != null">
            #{userId,jdbcType=VARCHAR},
        </if>
        <if test="wxUnionid != null">
            #{wxUnionid,jdbcType=VARCHAR},
        </if>
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用


```
                <if test="account != null">
                    #{account,jdbcType=VARCHAR},
                </if>
                <if test="password != null">
                    #{password,jdbcType=VARCHAR},
                </if>
                <if test="accountType != null">
                    #{accountType,jdbcType=INTEGER},
                </if>
                <if test="phone != null">
                    #{phone,jdbcType=VARCHAR},
                </if>
                <if test="headPic != null">
                    #{headPic,jdbcType=VARCHAR},
                </if>
                <if test="userName != null">
                    #{userName,jdbcType=VARCHAR},
                </if>
                <if test="nickName != null">
                    #{nickName,jdbcType=VARCHAR},
                </if>
                <if test="state != null">
                    #{state,jdbcType=INTEGER},
                </if>
                NOW()
            </trim>
        </insert>
        <select id="findOneUser" resultMap="BaseResultMap" parameterType="com.mz.live.dao.
entity.User">
            SELECT
            <include refid="Base_Column_List" />
            FROM t_user
            <where>
                <if test="wxUnionid != null">
                    AND wx_unionid = #{wxUnionid}
                </if>
                <if test="accountType != null">
                    AND account_type = #{accountType}
                </if>
                <if test="userId != null">
                    AND user_id = #{userId}
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用


```
            </if>
            <if test="password != null">
                AND password = #{password}
            </if>
            <if test="account != null">
                AND account = #{account}
            </if>
            <if test="nickName != null">
                AND nick_name = #{nickName}
            </if>
            <if test="userName != null">
                AND user_name = #{userName}
            </if>
            <if test="state != null">
                AND state = #{state}
            </if>
        </where>
        LIMIT 1
    </select>
```

2. Android 注册功能开发

向后台传递用户需要注册的账号（用户名：douxue1，密码：123456789），后台返回注册结果，成功注册后跳转至主界面，失败则提示失败的原因。

（1）实现步骤

① 打开 Android Studio，导入本教材提供的配套项目包，此项目包是一个没有实现具体功能的空工程。

② 在 TCRegisterMgr.java 文件中加入如下代码，来实现与后台传递数据。

```
public void registerByServer(final String account, final String password) {
    String url = "user/register?account=" + account + "&password=" + password;
    TCHttpEngine.getInstance().post(url, new TCHttpEngine.Listener() {
        @Override
        public void onResponse(int retCode, String retMsg, JSONObject retData) {
            if (null != retData) {
                // 保存自己服务器的用户信息
                UserInfo userInfo = TCUtils.JOToModel(retData, UserInfo.class);
                TCUserInfoMgr.getInstance().setGUserinfo(userInfo);
                pwdRegist(account, password);
            } else {
                if(null != mTCRegisterCallback)
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用


```
                    mTCRegisterCallback.onFailure(retCode, retMsg);
                }
            }
        });
    }
```

③ 在 TCRegisterActivity.java 文件中加入如下代码来调用服务。

```
private void attemptNormalRegist(String username, String password, String passwordVerify) {
    if (TCUtils.isUsernameVaild(username)) {
        if (TCUtils.isPasswordValid(password)) {
            if (password.equals(passwordVerify)) {
                if(TCUtils.isNetworkAvailable(this)) {
                    mPassword = password;
                    mTCRegisterMgr.registerByServer(username, password);
                    showOnLoading(true);
                } else {
                    Toast.makeText(getApplicationContext(), "当前无网络连接",
Toast.LENGTH_SHORT).show();
                }
            } else {
                showPasswordVerifyError("两次输入密码不一致");
            }
        } else {
            showPasswordVerifyError("密码长度应为 8～16 位");
        }
    } else {
        showRegistError("用户名不符合规范");
    }
}
```

④ 在注册按钮监听事件中调用此方法。

```
btnRegister.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //调用 normal 注册逻辑
        attemptNormalRegist(etRegister.getText().toString(),
                etPassword.getText().toString(), etPasswordVerify.getText().toString());
    }
});
```

⑤ 在 onSuccess()方法中加入如下代码，提示用户注册成功。

非卖品，仅供非商业用途或交流学习使用


```
        Toast.makeText(getApplicationContext(), "成功注册 " + identifier, Toast.LENGTH_SHORT).
show();
        showOnLoading(false);
        mTCRegisterMgr.removeTCRegisterCallback();
```

⑥ 在 onFailure()方法中加入如下代码，提示用户注册失败。

```
        Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_SHORT).show();
        showOnLoading(false);
```

（2）执行测试

① 执行效果。

单击快捷工具栏中的“运行”按钮▸，运行程序。程序开始显示一个注册界面，在“用户名”文本框中输入账号“douxue”，在“密码”和“确认密码”文本框中输入密码“123456789”，最后单击“注册”按钮。如图 4-2 所示。

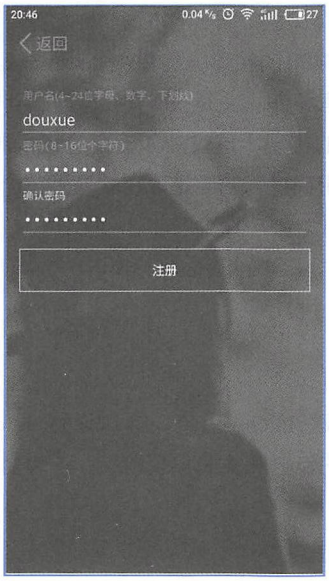


图 4-2
执行启动后的效果图

② 手动测试。

测试的场景（TestCase）如表 4-1 所示，可以编写单元测试或者自己手动测试。

测试场景	输入参数	预期结果
账号密码都正确	Username：douxue1 Password：123456789	注册成功
输入已经注册的账号	Username：douxue1 Password：123456789	注册失败，账号已存在
输入带有特殊符号的账号	Username：douxue\ Password：123456789	注册失败

表 4-1
测试用例

经过测试，不存在的账号能够正常注册，账号带有非法字符会被阻止。至此，本项目已经实现了注册功能的开发。

非卖品，仅供非商业用途或交流学习使用

项目实训

实训题目

完成注册时用户名和密码格式的约束。

实训目的

了解常规开发注册功能中应该禁止的用户名和密码。

实训内容

对用户注册的用户名和密码进行格式约束。

任务 4.2　实现用户登录模块

任务描述

Web 后台实现用户登录功能的接口，提供给 Android 程序调用，同时用于用户登录，用户登录成功后会把用户信息（包括主键 ID）一起返回到前端，方便业务操作，这里没有加入安全机制，只是简单的实现（建议加入安全验证机制，比如签名鉴权机制）。APP 先实现对用户输入的账号和密码进行判断，确定其是否符合规范，若符合规范则调用后台提供的接口进行登录。

知识学习

1. Activity 的跳转

Activity 的跳转需要创建 Intent 对象，通过设置 Intent 对象的参数指定要跳转的 Activity；通过设置 Activity 的包名和类名实现跳转，称为显式意图；通过指定动作实现跳转，称为隐式意图。

（1）显式意图

跳转至同一项目下的另一个 Activity，直接指定该 Activity 的字节码即可，代码如下。

```
Intent intent = new Intent();
intent.setClass(this, SecondActivity.class);
startActivity(intent);
```

跳转至其他应用中的 Activity，需要指定该应用的包名和该 Activity 的类名，代码如下。

```
Intent intent = new Intent();//启动系统自带的拨号器应用
intent.setClassName("com.android.dialer", "com.android.dialer.DialtactsActivity");
startActivity(intent);
```

（2）隐式意图

隐式意图跳转至指定 Activity 的代码如下。

非卖品，仅供非商业用途或交流学习使用


```
Intent intent = new Intent();//启动系统自带的拨号器应用
intent.setAction(Intent.ACTION_DIAL);
startActivity(intent);
```

要让一个 Activity 可以被隐式启动，需要在清单文件的 activity 节点中设置 intent-filter 子节点，代码如下。

```
<intent-filter >
    <action android:name="com.itheima.second"/>
    <data android:scheme="asd" android:mimeType="aa/bb"/>
    <category android:name="android.intent.category.DEFAULT"/></intent-filter>
```

参数说明如下：

- Action 标签表示动作（可以自定义，也可以使用系统自带的）。
- Data 标签表示数据（操作什么内容）。
- Category 标签表示类别（默认类别、机顶盒、车载电脑）。

隐式意图启动 Activity，需要为 Intent 设置以上 3 个属性，且值必须与该 Activity 在清单文件中对 3 个属性的定义匹配。

intent-filter 节点及其子节点都可以同时定义多个，隐式启动时只需与任意一个匹配即可。

（3）获取通过 setData 传递的数据

代码如下。

```
//获取启动此 Activity 的 Intent 对象
Intent intent = getIntent();
Uri uri = intent.getData();
```

（4）显式意图和隐式意图的应用场景

① 显式意图用于启动同一应用中的 Activity。

② 隐式意图用于启动不同应用中的 Activity。

如果系统中存在多个 Activity 的 intent-filter 同时与 Intent 匹配，那么系统会显示一个对话框，列出所有匹配的 Activity，由用户选择启动哪一个。

2. Activity 跳转时的数据传递

Activity 通过 Intent 启动时，可以通过 Intent 对象携带数据到目标 Activity，代码如下。

```
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtra("maleName", maleName);
intent.putExtra("femaleName", femaleName);
startActivity(intent);
```

在目标 Activity 中取出数据，代码如下。

```
Intent intent = getIntent();
String maleName = intent.getStringExtra("maleName");
String femaleName = intent.getStringExtra("femaleName");
```

非卖品，仅供非商业用途或交流学习使用

任务实现

微课 4-2
实现用户登录模块

用户登录功能具体流程如图 4-3 所示。

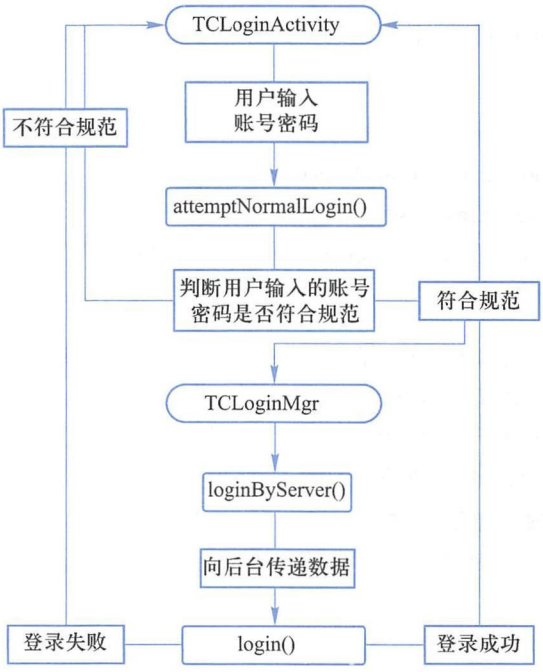


图 4-3
用户登录流程图

1. 后台用户登录接口开发

① 打开 Eclipse，导入本教材提供的配套项目包。

② 在 com.douxue.live.controller.app.use 中的 UserController.java 文件中实现登录功能的接口，添加的代码如下。

```
/**
 * 用户登录
 * @param account
 * @param password
 * @return
 *
 */
@RequestMapping(value = "login", method = RequestMethod.POST)
@ResponseBody
public RestResult login(@RequestParam("account") String account,
        @RequestParam("password") String password) {
    return userService.login(account, password);
}
```

③ 在 com.douxue.live.service.user 中的 UserService.java 文件中添加代码，代码如下。

```
/**
 * 用户登录
 * @param account
 * @param password
 * @return
 */
RestResult login(String account, String password);
```

④ 这是一个接口类，所以要在 com.douxue.live.service.user.impl 中的 UserServiceImpl.java 文件中来实现接口方法，代码如下。

```
/**
 * 用户登录
 */
@Override
public RestResult login(String account, String password) {
        User query = new User();
        query.setAccount(account);
query.setPassword(MD5Utils.encryptUserPassword(password));
        User user = userMapper.findOneUser(query);
        if (null == user) {
                return new RestResult(ErrorCode.ERROR_ACCOUNT_OR_PWD);
        }
        if(user.getState().intValue()== EnumUser.STATE_STOP.getValue()){
                return new RestResult(ErrorCode.ERROR_ACCOUNT_STOP);
        }
        user.setPassword(null);
        user.setWxUnionid(null);
        return new RestResult(user);
}
```

2. Android 登录功能开发

用户输入账号和密码（账号：douxue1，密码：123456789），进入主界面。

源码 4-2：实现用户登录模块

（1）实现步骤

① 打开 Android Studio，导入本教材提供的配套项目包。

② 在 TCLoginMgr.java 文件中加入如下代码，来传输数据。

```
public void loginByServer(final String username, final String password) {
    String url = "user/login?account=" + username + "&password=" + password;
    TCHttpEngine.getInstance().post(url, new TCHttpEngine.Listener() {
        @Override
```

```
                public void onResponse(int retCode, String retMsg, JSONObject retData) {
                    if (null != retData) {
                        // 保存自己服务器用户信息
                        UserInfo userInfo = TCUtils.JOToModel(retData, UserInfo.class);
                        TCUserInfoMgr.getInstance().setGUserinfo(userInfo);
                        pwdLogin(username, password);
                    } else {
                        if (null != mTCLoginCallback)
                            mTCLoginCallback.onFailure(retCode, retMsg);
                    }
                }
            });
        }
```

③ 在 TCLoginActivity.java 文件中加入如下代码来调用服务。

```
    public void attemptNormalLogin(String username, String password) {

        if (TCUtils.isUsernameVaild(username)) {
            if (TCUtils.isPasswordValid(password)) {
                if(TCUtils.isNetworkAvailable(this)) {
                    //调用 LoginHelper 进行普通登录
//                      mTCLoginMgr.pwdLogin(username, password);
                    mTCLoginMgr.loginByServer(username, password);
                } else {
                    Toast.makeText(getApplicationContext(), "当前无网络连接",
Toast.LENGTH_SHORT).show();
                    showOnLoading(false);
                }
            } else {
                showPasswordError("密码过短");
            }
        } else {
            showLoginError("用户名不符合规范");
        }
    }
```

④ 在登录按钮的监听事件中调用此方法。

```
//调用 normal 登入逻辑
showOnLoading(true);
attemptNormalLogin(etLogin.getText().toString(), etPassword.getText().toString());
```

⑤ 在 onSuccess()方法中加入如下代码，提示用户登入成功，并跳转至主页面。

```
        TCUserInfoMgr.getInstance().setUserId(mTCLoginMgr.getLastUserInfo().identifier,
new ITCU serInfoMgrListener() {
            @Override
            public void OnQueryUserInfo(int error, String errorMsg) {

            }

            @Override
            public void OnSetUserInfo(int error, String errorMsg) {
                if (0 != error) Toast.makeText(getApplicationContext(),"设置 User ID 失败" +
errorMsg,Toast.LENGTH_LONG).show();
            }
        });
        Toast.makeText(getApplicationContext(), "登录成功", Toast.LENGTH_SHORT).show();
        mTCLoginMgr.removeTCLoginCallback();
        showOnLoading(false);
        jumpToHomeActivity();
```

⑥ 加入 jumpToHomeActivity()方法来跳转主界面，代码如下。

```
    //跳转至主界面
    public void jumpToHomeActivity() {
        Intent intent = new Intent(this, TCMainActivity.class);
        startActivity(intent);
        finish();
    }
```

⑦ 在 onFailure()方法中加入如下代码，提示用户注册失败。

```
    Log.d(TAG, "Login Error errCode:" + i + " msg:" + s);
    showOnLoading(false);
    //被踢下线后弹窗显示被踢
    if (6208 == i) {
        TCUtils.showKickOutDialog(this);
    }
    Toast.makeText(getApplicationContext(), "登入失败" + s,
 Toast.LENGTH_SHORT).show();
```

（2）执行测试

① 执行效果。

单击快捷工具栏中的“运行”按钮▶，运行程序。在“用户名”文本框中输入账号“douxue1”，在“密码”文本框中输入密码“123456789”，最后单击“登录”按钮，如图 4-4 所示。

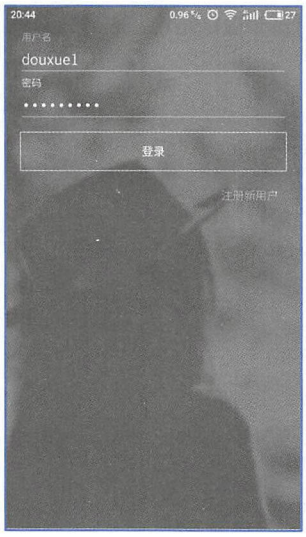


图 4-4
执行启动后的效果图

② 手动测试。

测试的场景（TestCase）如表 4-2 所示，可以编写单元测试或者自己手动测试。

表 4-2
测试用例

测试场景	输入参数	预期结果
账号密码都正确	Username：douxue1 Password：123456789	登入成功
输入错误的账号	Username：douxue2 Password：123456789	登入失败，账号或密码错误
输入错误的密码	Username：douxue\ Password：123456789	登入失败，账号或密码错误

经过测试，正确的账号和密码能够正常登入，错误的账号或密码无法登入。至此，本项目已经实现了登入功能的开发。

项目实训

实训题目

练习 Activity 跳转。

实训目的

1. 了解 Activity 的跳转方式。
2. 掌握 Activity 的跳转方法。

实训内容

开发 Demo，要求完成两个 Activity 之间的互相跳转。

任务 4.3　实现个人中心模块

任务描述

了解 Android Studio 导入 jar 包的方式，在导入 imsdk.jar 包用户登录之后通过 imsdk.jar

包中的方法获取用户信息。

知识学习

Android Studio 导入 jar 包

① 将 imsdk.jar 包复制到 jniLibs 文件夹中，如图 4-5 所示。

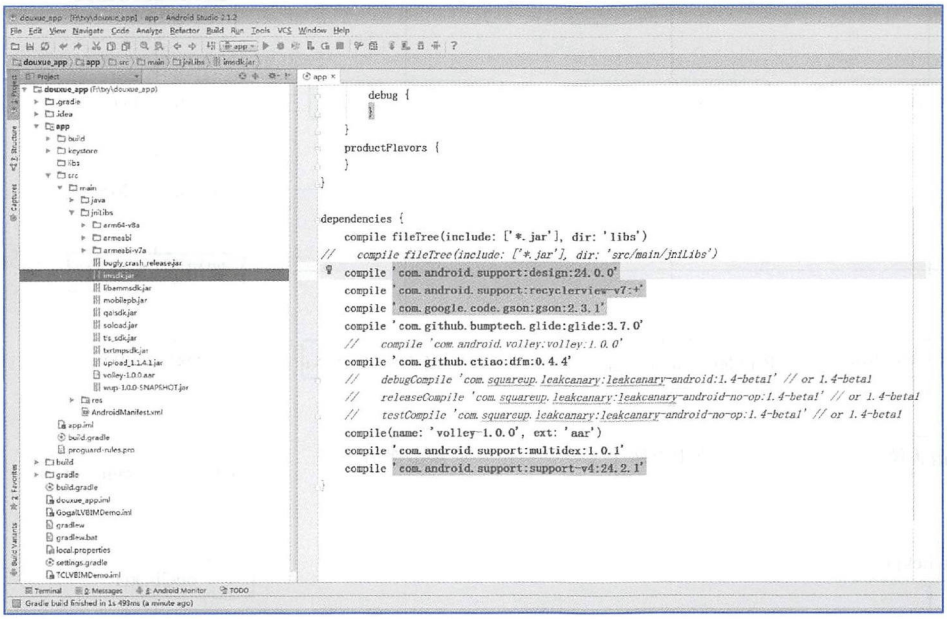


图 4-5
导入 JAR 包

② 选中 imsdk.jar，然后右击，在弹出的快捷菜单中选择“Add As Library”命令，完成导入，如图 4-6 所示。

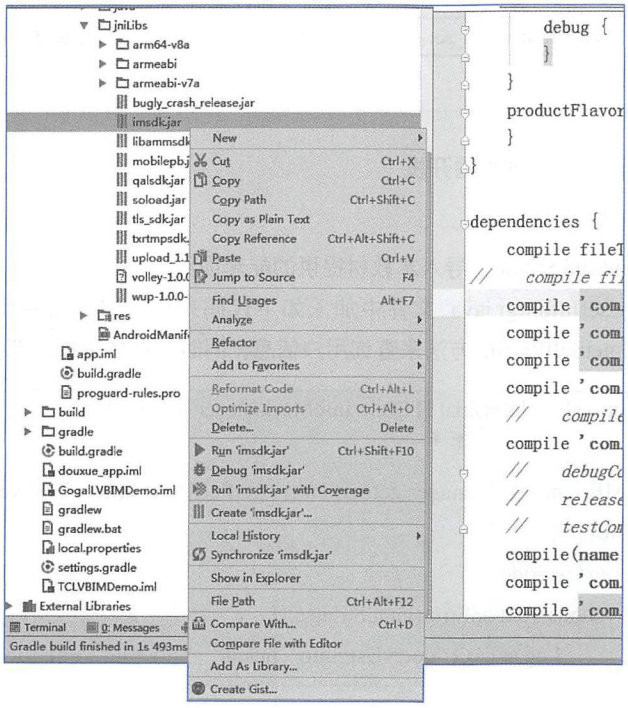


图 4-6
右键快捷菜单

任务实现

个人中心功能具体流程如图 4-7 所示。

微课 4-3
实现个人中心模块

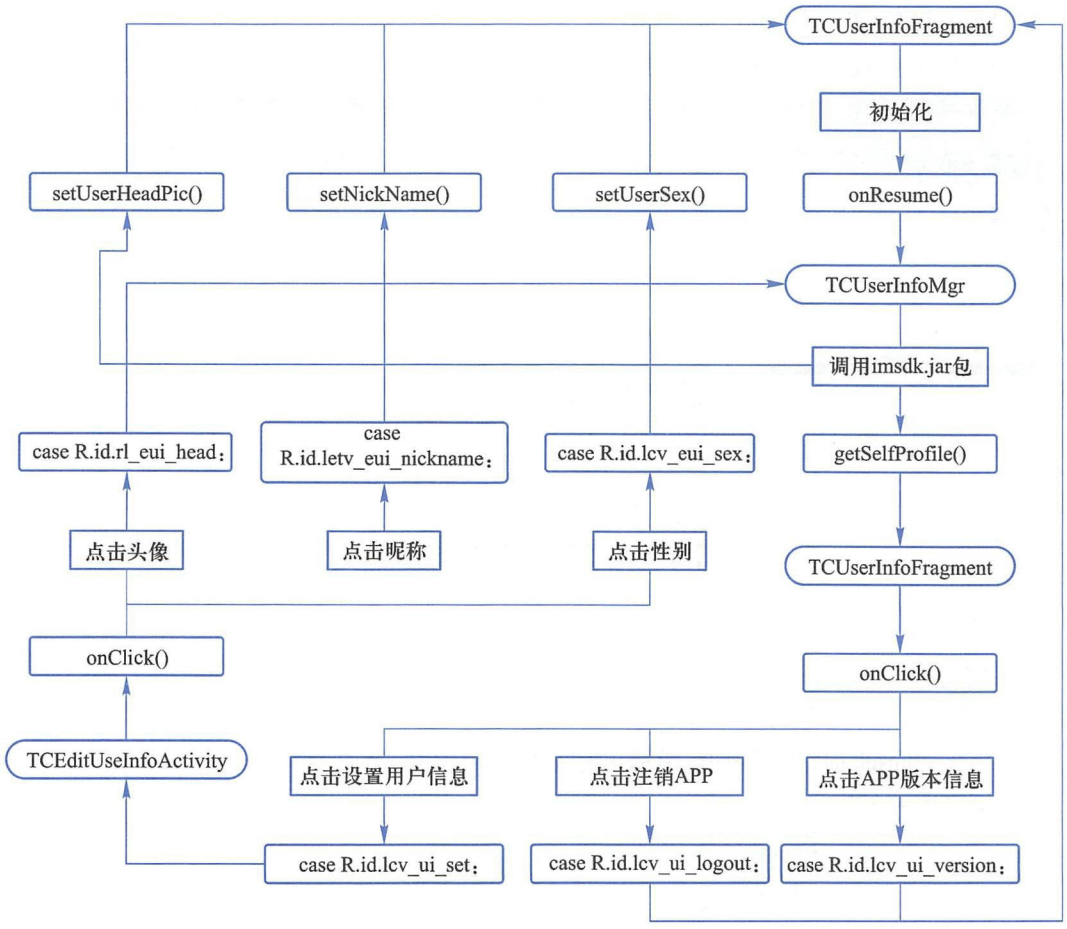


图 4-7
个人中心流程图

Android 个人中心功能的开发

源码 4-3:
实现个人中心模块

（1）实现步骤

① 打开 Android Studio，导入本教材提供的配套项目包。

② 在 TCUserInfoMgr.java 文件中加入如下代码来获获取并设置用户信息，调用 imsdk.jar 包中的 getSelfProfile 方法来查询用户信息，同时对结果设置回调：

```
public void queryUserInfo(final ITCUserInfoMgrListener listener){
    try {
        TIMFriendshipManager.getInstance().getSelfProfile(new TIMValueCallBack
<TIMUserProfile>() {
            @Override
            public void onError(int i, String s) {
                Log.e(TAG, "queryUserInfo  failed  , " + i + " : " + s);
```

```
                if (listener != null) {
                    listener.OnQueryUserInfo(i,s);
                }
            }

            @Override
            public void onSuccess(TIMUserProfile timUserProfile) {
                Log.e(TAG, "queryUserInfo    success!");

                mUserInfo.userId = timUserProfile.getIdentifier();
                if (TextUtils.isEmpty(timUserProfile.getNickName())) {
                    mUserInfo.nickname = timUserProfile.getIdentifier();
                } else {
                    mUserInfo.nickname = timUserProfile.getNickName();
                }
                mUserInfo.headPic = timUserProfile.getFaceUrl();
                mUserInfo.coverPic = timUserProfile.getSelfSignature();
                mUserInfo.sex = timUserProfile.getGender();
                mUserInfo.location = timUserProfile.getLocation();

                if (listener != null) {
                    listener.OnQueryUserInfo(0, null);
                }
            }
        });
    } catch (Exception e){
        e.printStackTrace();
    }

}
```

③ 通过查找出来的信息给 bean 类中的 userid 赋值，并使用 ID 向服务器查询用户信息。

```
public void setUserId(final String userId, final ITCUserInfoMgrListener listener) {
    try {
        queryUserInfo(new ITCUserInfoMgrListener() {
            @Override
            public void OnQueryUserInfo(int error, String errorMsg) {
                if (0 == error) {
                    mUserInfo.userId = userId;
                } else {
                    Log.e(TAG,"setUserId failed:" + error + "," + errorMsg);
```


版权所有，严禁传播，违者自负法律责任！

```
                }
                if (null != listener)
                    listener.OnSetUserInfo(error, errorMsg);
            }

            @Override
            public void OnSetUserInfo(int error, String errorMsg) {

            }
        });
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

④ 通过调用 imsdk.jar 包中的 setNickName 方法给服务器中的昵称赋值，同时修改本地 bean 类中的昵称，同时给结果设置回调。

```
public void setUserNickName(final String nickName, final ITCUserInfoMgrListener listener){
    if (mUserInfo.nickname.equals(nickName)) {
        if (null != listener) listener.OnSetUserInfo(0, null);
        return;
    }

    TIMFriendshipManager.getInstance().setNickName(nickName, new TIMCallBack() {
        @Override
        public void onError(int i, String s) {
            Log.e(TAG,"setUserNickName failed:" + i + "," + s);
            if (null != listener)listener.OnSetUserInfo(i,s);
        }

        @Override
        public void onSuccess() {
            mUserInfo.nickname = nickName;
            if (null != listener) listener.OnSetUserInfo(0,null);
        }
    });
}
```

⑤ 通过调用 imsdk.jar 包中的 setGender 方法给服务器中的性别赋值，同时修改本地 bean 类中的性别，同时给结果设置回调。

版权所有，严禁传播，违者自负法律责任！

```
public void setUserSex(final TIMFriendGenderType sex, final ITCUserInfoMgrListener listener){
    if (mUserInfo.sex == sex) {
        if (null != listener) listener.OnSetUserInfo(0,null);
        return;
    }

    TIMFriendshipManager.getInstance().setGender(sex, new TIMCallBack() {
        @Override
        public void onError(int i, String s) {
            Log.e(TAG,"setUserSex failed:" + i + "," + s);
            if (null != listener) listener.OnSetUserInfo(i,s);
        }

        @Override
        public void onSuccess() {
            mUserInfo.sex = sex;
            if (null != listener) listener.OnSetUserInfo(0,null);
        }
    });
}
```

⑥ 通过调用 imsdk.jar 包中的 setFaceUrl 方法将图片上传至服务器，服务器返回上传后的 URL 来实现，同时设置结果回调。

```
public void setUserHeadPic(final String url, final ITCUserInfoMgrListener listener){
    TIMFriendshipManager.getInstance().setFaceUrl(url, new TIMCallBack() {
        @Override
        public void onError(int i, String s) {
            Log.e(TAG,"setUserHeadPic failed:" + i + "," + s);
            if (null != listener) listener.OnSetUserInfo(i,s);
        }

        @Override
        public void onSuccess() {
            mUserInfo.headPic = url;
            if (null != listener) listener.OnSetUserInfo(0, null);
        }
    });
}
```


版权所有，严禁传播，违者自负法律责任！

⑦ 在 TCUserInfoFragment.java 中加入如下代码调用 TCUserInfoMgr 类中的方法。

⑧ 在 onResume()方法中加入如下代码，在页面展示前，更新用户信息。

```
TCUserInfoMgr.getInstance().queryUserInfo(new ITCUserInfoMgrListener() {
    @Override
    public void OnQueryUserInfo(int error, String errorMsg) {
        if (0 == error) {
            mNickName.setText(TCUserInfoMgr.getInstance().getNickname());
            mUserId.setText("ID:" + TCUserInfoMgr.getInstance().getUserId());
            TCUtils.showPicWithUrl(getActivity(), mHeadPic, TCUserInfoMgr.getInstance().
getHeadPic(), R.drawable.face);
        }
    }

    @Override
    public void OnSetUserInfo(int error, String errorMsg) {

    }
});
```

⑨ 为个人中心界面上的 3 个按钮设置响应事件。

```
@Override
public void onClick(View view) {
    switch (view.getId()) {
        case R.id.lcv_ui_set: //设置用户信息
            enterEditUserInfo();
            break;
        case R.id.lcv_ui_logout: //注销 APP
            TCLoginMgr.getInstance().logout();
            Intent intent = new Intent(getContext(), TCLoginActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
            getActivity().finish();
            break;
        case R.id.lcv_ui_version: //显示 APP SDK 的版本信息
            showSDKVersion();
            break;
    }
}
```

⑩ 跳转用户信息设置页面，代码如下，其中 TCEditUseInfoActivity.class 用来设置用户信息，此类中调用了 TCUserInfoMgr 类中的方法对用户信息进行了设置（代码在工程中附带）。

版权所有，严禁传播，违者自负法律责任！

```
private void enterEditUserInfo() {
    try {
        Intent intent = new Intent(getContext(), TCEditUseInfoActivity.class);
        startActivity(intent);
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

⑪ 展示 APP 版本信息，代码如下。

```
private String getAppVersion() {
    PackageManager packageManager = getActivity().getPackageManager();
    PackageInfo packInfo;
    String version = "";
    try {
        packInfo = packageManager.getPackageInfo(getActivity().getPackageName(), 0);
        version = packInfo.versionName;
        Toast.makeText(getActivity(),version, Toast.LENGTH_SHORT).show();
    } catch (PackageManager.NameNotFoundException e) {
        e.printStackTrace();
    }
    return version;
}
```

（2）执行测试

① 执行效果。

单击快捷工具栏中的“运行”按钮▶，运行程序。使用用户名“douxue1”、密码“123456789”登录，进入主界面单击“个人中心”按钮，如图 4-8 所示。

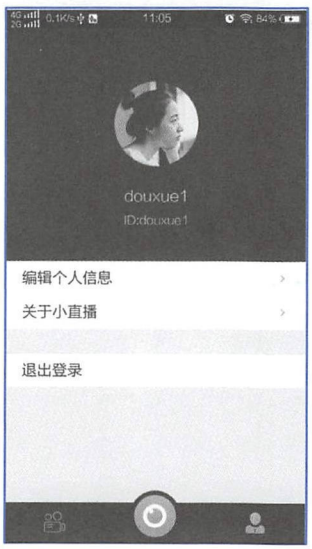


图 4-8
执行启动后的效果图

版权所有，严禁传播，违者自负法律责任！

② 手动测试。

测试的场景（TestCase）如表 4-3 所示，可以编写单元测试或者自己手动测试。

表 4-3
测试用例

测试场景	输入参数	预期结果
修改昵称	将昵称修改为 douxue2	修改成功
单击关于小直播	—	弹出提示框 APP 版本
单击退出登录	—	跳转至登录界面

经过测试，所有个人中心的功能都能正常使用。至此，本项目已经实现了个人中心的开发。

项目实训

实训题目

练习 Android Studio 的几种导入/导出包方式。

实训目的

1. 了解 Android Studio 可以导入什么类型的包。
2. 掌握 Android Studio 的导入/导出包的方法。

实训内容

在网络上查找相关资料来了解 Android Studio 导入/导出包并在 Android Studio 进行练习。

单元小结

本单元演示完成了整个用户管理模块的开发，包括用户注册、登录进入主界面、进入个人中心。本单元重点讲述如何进行前后台通信，同时涉及 MD5 加密技术的使用及 Android Studio 中第三方 jar 包的使用，相信读者在学习了本单元之后对 MD5 会有进一步的理解。

版权所有，严禁传播，违者自负法律责任！

单元 5 直播模块

学习目标

PPT5:
直播模块

知识目标

- 了解后台的实现方法。
- 了解如何调用后台接口。
- 了解如何开启推流服务和结束推流服务。

技能目标

- 掌握开启直播功能的实现方法。
- 掌握直播列表功能的实现方法。
- 掌握观看直播功能的实现方法。

版权所有，严禁传播，违者自负法律责任！

学习情境

斗学直播 APP 在小赵的努力下已经完成了 APP 的登录注册等功能，用户可以登录系统查看和修改自己的个人信息，但是不能进行直播。小赵计划接下来完善直播 APP 的直播模块，要求直播 APP 能支持用户开启直播，并在直播列表中显示当前直播房间，其他用户可以进入观看直播。

任务 5.1　实现开启直播模块

任务描述

在 Web 后台实现创建直播、开启推流和结束推流 3 个功能的接口，提供给 Android 程序调用，APP 先实现对直播间的设置，其中涉及封面的上传、直播标题的设置、定位的获取、选择直播的方式。设置完成后调用后台提供的接口完成直播间的推流工作。

知识学习

APP 在发起直播前首先会通过一条协议向服务器请求创建一个直播间。

1. APP 向后台服务发起请求

最关键的信息是用户账号 ID，同时最好再附上直播的标题、地理位置和直播封面等信息，直播后台会用这些信息创建一个直播间。

2. 后台服务响应 APP 请求

后台服务的响应回包信息包含两种情况：一是允许直播，则可以把推流 URL 等信息返回给 APP；另一种情况是给予拒绝并返回错误原因信息。

如图 5-1 所示是允许创建直播响应返回的一条标准的带防盗链签名的推流 URL。

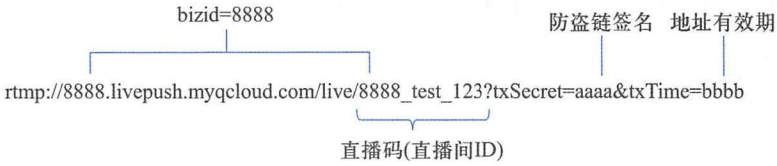


图 5-1
标准带防盗链签名的推流

3. 修改状态

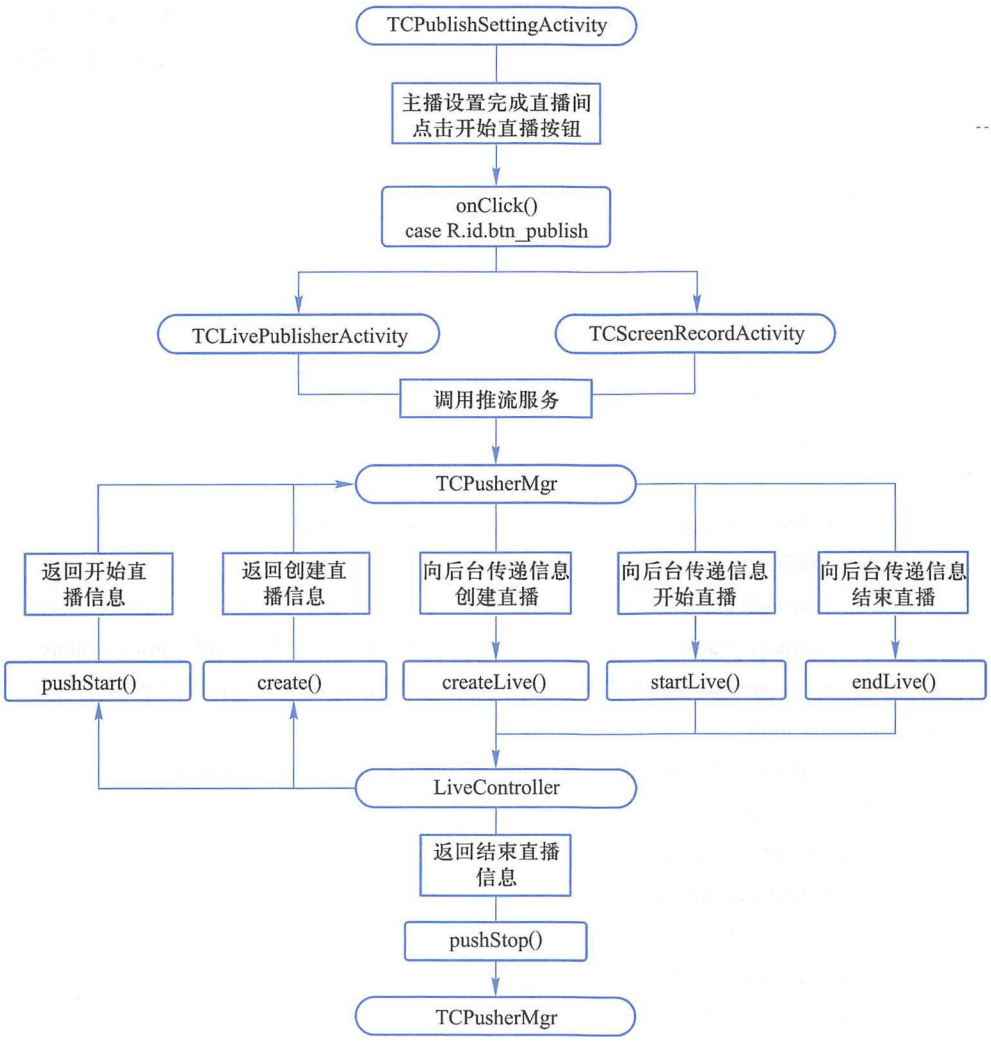
新创建的直播间不要直接设置状态为“直播中”，因为主播并不一定能够成功推流，这里可能的原因有：推流用的端口 1935 被所处网络的安全防火墙禁用了，或者是主播的 APP 刚刚安装，在看到摄像头权限申请时误点了拒绝。

正确的解决方法应该是让 APP 在成功收到 RTMP SDK 的 PUSH_BEGIN 事件后确定推流成功，之后再通过一条协议通知后台服务把直播间状态改为“直播中”，或者等待腾讯云的推流事件通知（腾讯云在收到某个直播流的视频数据后会发送一个事件通知到配置的 URL 上）后再改状态。

版权所有，严禁传播，违者自负法律责任！

任务实现

开启直播功能具体流程图如图 5-2 所示。



微课 5-1
实现开启直播模块

图 5-2
开启直播流程图

1. 后台开启直播接口开发

(1) 创建直播

创建一个新直播，参考腾讯云直播管理说明文档生成直播推/拉流地址，并保存数据库，这时候直播状态为“初始化”，保存并将直播信息返回给 APP 后，APP 拿到推流地址后就可以开始直播推流与观看拉流。

(2) 实现步骤

首先，APP 客户端发送一个创建直播请求，将创建直播用到的用户 ID、群组 ID、标题、封面等信息传递给后台，后台利用这些信息创建出一个 Live 对象，保存直播的信息。最后调用 LiveServiceImpl 类中的 createLive()方法创建直播，并将直播的信息返回给 APP 客户端。

版权所有，严禁传播，违者自负法律责任！

源码 5-1：
实现开启直播模块

开始和结束直播需要一个 streamId，也即 Live 对象的 ID，根据这个 streamId 可以得到 Live 对象，然后开始推流。

① 打开 Eclipse，导入本教材提供的配套项目包。

② 在 com.douxue.live.controller.app.live.LiveController.java 文件中添加以下代码，完成创建直播、开始直播与结束直播的功能实现，用于接受 APP 客户端发送请求所带的参数。

```
@Autowired
private LiveService liveService;
/*
 * 创建直播
 * @param userId
 * @param groupId
 * @param title
 * @return
 */
@RequestMapping(value = "create", method = RequestMethod.POST)
@ResponseBody
public RestResult create(
        @RequestParam("userId") String userId, @RequestParam("groupId") String groupId,
        @RequestParam("title") String title,@RequestParam(value = "location", required =
false) String location,
        @RequestParam(value = "coverPic", required = false) String coverPic) {
    Live live = new Live();
    live.setUserId(userId);
    live.setGroupId(groupId);
    live.setTitle(title);
    live.setLocation(location);
    live.setCoverPic(coverPic);
    return liveService.createLive(live);
}
```

③ 开启直播时需要添加的 control 层代码，接受客户端发送的直播间 ID，对其数据库中的状态进行修改，完成对直播的开启和关闭。

```
/**
 * 开始直播
 * @param streamId
 * @return
 *
 */
@RequestMapping(value = "push/start", method = RequestMethod.POST)
```

```
@ResponseBody
public RestResult pushStart(@RequestParam("streamId") String streamId) {
    Live live = liveService.pushStart(streamId);
    if (null != live) {
        return new RestResult(live);
    }
    return RestResult.FAIL;
}
/**
 * 结束直播
 * @param streamId
 * @return
 *
 */
@RequestMapping(value = "push/stop", method = RequestMethod.POST)
@ResponseBody
public RestResult pushStop(@RequestParam("streamId") String streamId) {
    Live live = liveService.pushStop(streamId);
    if (null != live) {
        return new RestResult(live);
    }
    return RestResult.FAIL;
}
```

④ 在 com.douxue.live.service.live 中的 LiveService.java 文件中实现 service 中的功能，代码如下。

```
/**
 * 创建直播
 * @param live
 * @return
 *
 */
public RestResult createLive(Live live);

/**
 * 直播开始推流
 * @param streamId
 * @return
 *
 */
public Live pushStart(String streamId);
```

```
    /**
     * 直播结束推流
     * @param streamId
     * @return
     *
     */
    public Live pushStop(String streamId);
```

⑤ 在 com.douxue.live.service.live.impl 中的 LiveServiceImpl.java 文件中实现 service 接口中的创建直播，代码如下。

```
    private static final long TIMEOUT = 1800;
    @Value("${live_appid}")
    private String LIVE_APP_ID;
    @Value("${live_bizid}")
    private String LIVE_BIZID;

    @Value("${live_common_access_url}")
    private String LIVE_COMMON_ACCESS_URL;

    @Autowired
    private LiveApiTool liveApiTool;
    @Autowired
    private UserMapper userMapper;
    @Autowired
    private LiveMapper liveMapper;

    @Override
    @Transactional
    public RestResult createLive(Live create) {
        User user = userMapper.selectByPrimaryKey(create.getUserId());
        if (null == user) {
            return new RestResult(ErrorCode.ERROR_TARGET_USER_NO_EXIST);
        }
        // 创建直播
        String streamId = LIVE_BIZID + "_" + UUIDUtils.getUUID(); // 生成直播 ID
        create.setStreamId(streamId);
        create.setCreateTime(new Date());
        create.setState(EnumLive.STATE_CREATE.getValue());// 设置直播为初始化状态
        // 拼接推流地址
        create.setPushUrl(liveApiTool.getPushRtmpUrl(streamId, System.currentTimeMillis() /
1000 + TIMEOUT));
```

```
    // 拼接拉流地址
    create.setPlayRtmpUrl(liveApiTool.getPlayRtmpUrl(streamId));
    create.setPlayFlvUrl(liveApiTool.getPlayFlvURL(streamId));
    create.setPlayHlsUrl(liveApiTool.getPlayHlsUrl(streamId));
    if (liveMapper.insertSelective(create) > 0) {
        Live live = liveMapper.selectByPrimaryKey(streamId);
        return new RestResult(live);
    }
    return RestResult.FAIL;
}
```

⑥ 添加修改直播间的开始推流和结束推流的具体方法，代码如下。

```
@Override
public Live pushStart(String streamId) {
    Live live = liveMapper.selectByPrimaryKey(streamId);
    if (null != live) {
        Live changeLive = new Live();
        changeLive.setStreamId(live.getStreamId());
        changeLive.setState(EnumLive.STATE_PUSH.getValue());
        changeLive.setStartTime(new Date());
        if (liveMapper.updateByPrimaryKeySelective(changeLive) > 0) {
            return liveMapper.selectByPrimaryKey(streamId);
        }
    }
    return null;
}

@Override
public Live pushStop(String streamId) {
    Live live = liveMapper.selectByPrimaryKey(streamId);
    if (null != live) {
        Live changeLive = new Live();
        changeLive.setStreamId(live.getStreamId());
        changeLive.setState(EnumLive.STATE_STOP.getValue());
        if (liveMapper.updateByPrimaryKeySelective(changeLive) > 0) {
            return liveMapper.selectByPrimaryKey(streamId);
        }
    }
    return null;
}
```

⑦ 在 com.douxue.live.service.live.impl 中的 LiveApiTool.java 文件中完成直播播放需

要的代码，具体代码如下。

```
@Value("${live_appid}")
private String LIVE_APP_ID;
@Value("${live_bizid}")
private String LIVE_BIZID;
@Value("${live_push_safe_key}")
private String LIVE_PUSH_SAFE_KEY;
@Value("${live_api_sign_key}")
private String LIVE_API_SIGN_KEY;

@Value("${live_push_rtmp_url}")
private String LIVE_PUSH_RTMP_URL;
@Value("${live_play_rtmp_url}")
private String LIVE_PLAY_RTMP_URL;
@Value("${live_play_flv_url}")
private String LIVE_PLAY_FLV_URL;
@Value("${live_play_hls_url}")
private String LIVE_PLAY_HLS_URL;
@Value("${live_record}")
private String LIVE_RECORD;
```

⑧ 在创建直播时生成防盗链的具体代码如下。

```
public String getSafeUrl(String key, String streamId, long txTime) {
    String input = new StringBuilder().append(key).append(streamId).append(Long.toHexString
(txTime).toUpperCase()).toString();
    String txSecret = null;
    txSecret = MD5Utils.encrypt(input);
    return txSecret == null ? "" : new StringBuilder().append("&txSecret=").append(txSecret).
append("&").append("txTime=").append(Long.toHexString(txTime).toUpperCase()).toString();
}
```

⑨ 在创建直播时获取拼接推流地址的具体代码如下。

```
public String getPushRtmpUrl(String streamId, long txTime) {
    StringBuffer pushUrl = new StringBuffer();
    pushUrl.append(String.format(LIVE_PUSH_RTMP_URL, LIVE_BIZID, streamId, LIVE_
BIZID));
    // 回放生成格式
    if (null != LIVE_RECORD && !"".equals(LIVE_RECORD)) {
        pushUrl.append("&record=").append(LIVE_RECORD);
    }
```

```
        pushUrl.append(getSafeUrl(LIVE_PUSH_SAFE_KEY, streamId, txTime));
        return pushUrl.toString();
    }
```

⑩ 在创建直播时获取播放拉流地址、获取 FLV 播放地址、获取 HLS 播放地址的具体代码如下。

```
    public String getPlayRtmpUrl(String streamId) {
        return String.format(LIVE_PLAY_RTMP_URL, LIVE_BIZID, streamId);
    }
    public String getPlayFlvURL(String streamId) {
        return String.format(LIVE_PLAY_FLV_URL, LIVE_BIZID, streamId);
    }
    public String getPlayHlsUrl(String streamId) {
        return String.format(LIVE_PLAY_HLS_URL, LIVE_BIZID, streamId);
    }
```

⑪ 在 com.douxue.live.dao.mapper 中的 UserMapper.java 文件中添加代码，代码如下。

```
User selectByPrimaryKey(String userId);
```

⑫ 在 userMapper.xml 中实现查询的方法的代码如下。

```
<select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.String">
    select
    <include refid="Base_Column_List" />
    from t_user
    where user_id = #{userId,jdbcType=VARCHAR}
</select>
```

⑬ 在 com.douxue.live.dao.mapper 中的 LiveMapper.java 类中添加插入、查询等接口，代码如下。

```
int insertSelective(Live record);
Live selectByPrimaryKey(String streamId);
int updateByPrimaryKeySelective(Live record);
```

⑭ 在 LiveMapper.xml 中实现 LiveMapper.java 中的方法，代码如下。

```
  <insert id="insertSelective" parameterType="com.douxue.live.dao.entity.Live" >
    insert into t_live
    <trim prefix="(" suffix=")" suffixOverrides="," >
      <if test="streamId != null" >
        stream_id,
      </if>
      <if test="userId != null" >
```

```
    user_id,
</if>
<if test="groupId != null" >
    group_id,
</if>
<if test="title != null" >
    title,
</if>
<if test="coverPic != null" >
    cover_pic,
</if>
<if test="location != null" >
    location,
</if>
<if test="pushUrl != null" >
    push_url,
</if>
<if test="playRtmpUrl != null" >
    play_rtmp_url,
</if>
<if test="playFlvUrl != null" >
    play_flv_url,
</if>
<if test="playHlsUrl != null" >
    play_hls_url,
</if>
<if test="playUrl != null" >
    play_url,
</if>
<if test="state != null" >
    state,
</if>
<if test="duration != null" >
    duration,
</if>
<if test="startTime != null" >
    start_time,
</if>
<if test="createTime != null" >
    create_time,
</if>
```

```
    </trim>
    <trim prefix="values (" suffix=")" suffixOverrides="," >
      <if test="streamId != null" >
        #{streamId,jdbcType=VARCHAR},
      </if>
      <if test="userId != null" >
        #{userId,jdbcType=VARCHAR},
      </if>
      <if test="groupId != null" >
        #{groupId,jdbcType=VARCHAR},
      </if>
      <if test="title != null" >
        #{title,jdbcType=VARCHAR},
      </if>
      <if test="coverPic != null" >
        #{coverPic,jdbcType=VARCHAR},
      </if>
      <if test="location != null" >
        #{location,jdbcType=VARCHAR},
      </if>
      <if test="pushUrl != null" >
        #{pushUrl,jdbcType=VARCHAR},
      </if>
      <if test="playRtmpUrl != null" >
        #{playRtmpUrl,jdbcType=VARCHAR},
      </if>
      <if test="playFlvUrl != null" >
        #{playFlvUrl,jdbcType=VARCHAR},
      </if>
      <if test="playHlsUrl != null" >
        #{playHlsUrl,jdbcType=VARCHAR},
      </if>
      <if test="playUrl != null" >
        #{playUrl,jdbcType=VARCHAR},
      </if>
      <if test="state != null" >
        #{state,jdbcType=INTEGER},
      </if>
      <if test="duration != null" >
        #{duration,jdbcType=BIGINT},
      </if>
```

```
            <if test="startTime != null" >
              #{startTime,jdbcType=TIMESTAMP},
            </if>
            <if test="createTime != null" >
              #{createTime,jdbcType=TIMESTAMP},
            </if>
          </trim>
          </insert>

          <select id="selectByPrimaryKey"resultMap="BaseResultMap" parameterType=
"java.lang.String">
            select
            <include refid="Base_Column_List" />
            from t_live
            where stream_id = #{streamId,jdbcType=VARCHAR}
          </select>

        <update id="updateByPrimaryKeySelective"parameterType="com.douxue.live.dao.entity.Live" >
          update t_live
        <set >
          <if test="userId != null" >
            user_id = #{userId,jdbcType=VARCHAR},
          </if>
          <if test="groupId != null" >
            group_id = #{groupId,jdbcType=VARCHAR},
          </if>
          <if test="title != null" >
            title = #{title,jdbcType=VARCHAR},
          </if>
          <if test="coverPic != null" >
            cover_pic = #{coverPic,jdbcType=VARCHAR},
          </if>
          <if test="location != null" >
            location = #{location,jdbcType=VARCHAR},
          </if>
          <if test="pushUrl != null" >
            push_url = #{pushUrl,jdbcType=VARCHAR},
          </if>
          <if test="playRtmpUrl != null" >
            play_rtmp_url = #{playRtmpUrl,jdbcType=VARCHAR},
          </if>
```

```
        <if test="playFlvUrl != null" >
          play_flv_url = #{playFlvUrl,jdbcType=VARCHAR},
        </if>
        <if test="playHlsUrl != null" >
          play_hls_url = #{playHlsUrl,jdbcType=VARCHAR},
        </if>
        <if test="playUrl != null" >
          play_url = #{playUrl,jdbcType=VARCHAR},
        </if>
        <if test="state != null" >
          state = #{state,jdbcType=INTEGER},
        </if>
        <if test="duration != null" >
          duration = #{duration,jdbcType=BIGINT},
        </if>
        <if test="startTime != null" >
          start_time = #{startTime,jdbcType=TIMESTAMP},
        </if>
        <if test="createTime != null" >
          create_time = #{createTime,jdbcType=TIMESTAMP},
        </if>
      </set>
      where stream_id = #{streamId,jdbcType=VARCHAR}
    </update>
```

2. APP 开启直播功能开发

(1) 实现步骤

① 打开 Android Studio，导入本教材提供的配套项目包。

② 在开启直播前，需要对直播间封面、定位、标题、直播方式进行设置。在 TCPublishSettingActivity.java 类中加入如下代码，实现对直播的设置。

对于返回按钮的监听事件，代码如下。

```
case R.id.btn_cancel:
    finish();
    break;
```

对于封面按钮的监听事件，代码如下。

```
case R.id.cover:
    mPicChsDialog.show();
    break;
```

选择封面的图片选择对话框，代码如下。

非卖品，仅供非商业用途或交流学习使用


```
private void initPhotoDialog() {
    mPicChsDialog = new Dialog(this, R.style.floag_dialog);
    mPicChsDialog.setContentView(R.layout.dialog_pic_choose);
    WindowManager windowManager = getWindowManager();
    Display display = windowManager.getDefaultDisplay();
    Window dlgwin = mPicChsDialog.getWindow();
    WindowManager.LayoutParams lp = dlgwin.getAttributes();
    dlgwin.setGravity(Gravity.BOTTOM);
    lp.width = (int) (display.getWidth()); //设置宽度
    mPicChsDialog.getWindow().setAttributes(lp);
    TextView camera = (TextView) mPicChsDialog.findViewById(R.id.chos_camera);
    TextView picLib = (TextView) mPicChsDialog.findViewById(R.id.pic_lib);
    TextView cancel = (TextView) mPicChsDialog.findViewById(R.id.btn_cancel);
    camera.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            getPicFrom(CAPTURE_IMAGE_CAMERA);
            mPicChsDialog.dismiss();
        }
    });

    picLib.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            getPicFrom(IMAGE_STORE);
            mPicChsDialog.dismiss();
        }
    });
    cancel.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mPicChsDialog.dismiss();
        }
    });
}
```

根据选择的图片来源展示不同路径下的图片，代码如下。

```
private void getPicFrom(int type) {
    if (!mPermission) {
        Toast.makeText(this, getString(R.string.tip_no_permission), Toast.LENGTH_
SHORT).show();
```

非卖品，仅供非商业用途或交流学习使用


```
            return;
        }

        switch (type) {
            case CAPTURE_IMAGE_CAMERA:
                fileUri = createCoverUri("");
                Intent intent_photo = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                intent_photo.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
                startActivityForResult(intent_photo, CAPTURE_IMAGE_CAMERA);
                break;
            case IMAGE_STORE:
                fileUri = createCoverUri("_select");
                Intent intent_album = new Intent("android.intent.action.GET_CONTENT");
                intent_album.setType("image/*");
                startActivityForResult(intent_album, IMAGE_STORE);
                break;
        }
    }
```

在回调方法中，加入如下代码获取所选择图片的路径并上传。

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK) {
        switch (requestCode) {
            case CAPTURE_IMAGE_CAMERA:
                startPhotoZoom(fileUri);
                break;
            case IMAGE_STORE:
                String path = TCUtils.getPath(this, data.getData());
                if (null != path) {
                    Log.d(TAG, "startPhotoZoom->path:" + path);
                    File file = new File(path);
                    startPhotoZoom(Uri.fromFile(file));
                }
                break;
        }
    }
}
```

自定义切图工具对图片进行切图，代码如下。

```
public void startPhotoZoom(Uri uri) {
    cropUri = createCoverUri("_crop");
```

非卖品，仅供非商业用途或交流学习使用


```
        Intent intent = new Intent("com.android.camera.action.CROP");
        intent.setDataAndType(uri, "image/*");
        intent.putExtra("crop", "true");
        intent.putExtra("aspectX", 750);
        intent.putExtra("aspectY", 550);
        intent.putExtra("outputX", 750);
        intent.putExtra("outputY", 550);
        intent.putExtra("scale", true);
        intent.putExtra("return-data", false);
        intent.putExtra(MediaStore.EXTRA_OUTPUT, cropUri);
        intent.putExtra("outputFormat", Bitmap.CompressFormat.JPEG.toString());
        startActivityForResult(intent, CROP_CHOOSE);
    }
```

在回调方法中加入如下代码来获取切图后的图片并上传。

```
    case CROP_CHOOSE:
        tvPicTip.setVisibility(View.GONE);
        cover.setImageBitmap(null);
        cover.setImageURI(cropUri);
        mUploading = true;
        mUploadHelper.uploadCover(cropUri.getPath());
        break;
```

通过此方法上传封面，代码如下。

```
    @Override
    public void onUploadResult(int code, String url) {
        if (0 == code) {
            TCUserInfoMgr.getInstance().setUserCoverPic(url, new ITCUserInfoMgrListener() {
                @Override
                public void OnQueryUserInfo(int error, String errorMsg) {
                }
                @Override
                public void OnSetUserInfo(int error, String errorMsg) {
                }
            });
            Toast.makeText(this, "上传封面成功", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "上传封面失败，错误码"+code, Toast.LENGTH_SHORT).show();
        }
        mUploading = false;
    }
```

非卖品，仅供非商业用途或交流学习使用

设置是否显示地理位置的代码如下。

```
case R.id.btn_lbs:
    if (btnLBS.getChecked()) {
        btnLBS.setChecked(false, true);
        tvLBS.setText(R.string.text_live_close_lbs);
    } else {
        btnLBS.setChecked(true, true);
        tvLBS.setText(R.string.text_live_location);
        if (TCLocationHelper.checkLocationPermission(this)) {
            if (!TCLocationHelper.getMyLocation(this, this)) {
                tvLBS.setText(getString(R.string.text_live_lbs_fail));
                //Toast.makeText(getApplicationContext(), "定位失败，请查看是否打开
GPS", Toast.LENGTH_SHORT).show();
                btnLBS.setChecked(false, false);
            }
        }
    }
    break;
```

申请定位权限，代码如下。

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    switch (requestCode) {
        case TCConstants.LOCATION_PERMISSION_REQ_CODE:
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                if (!TCLocationHelper.getMyLocation(this, this)) {
                    tvLBS.setText(getString(R.string.text_live_lbs_fail));
                    btnLBS.setChecked(false, false);
                }
            }
            break;
        case TCConstants.WRITE_PERMISSION_REQ_CODE:
            for (int ret : grantResults) {
                if (ret != PackageManager.PERMISSION_GRANTED) {
                    return;
                }
            }
            mPermission = true;
```

非卖品，仅供非商业用途或交流学习使用


```
                    break;
                default:
                    break;
            }
        }
```

设置位置的代码如下。

```
        @Override
        public void onLocationChanged(int code, double lat1, double long1, String location) {
            if (btnLBS.getChecked()) {
                if (0 == code) {
                    tvLBS.setText(location);
                    TCUserInfoMgr.getInstance().setLocation(location, lat1, long1, new
ITCUserInfoMgrListener() {
                        @Override
                        public void OnQueryUserInfo(int error, String errorMsg) {
                        }
                        @Override
                        public void OnSetUserInfo(int error, String errorMsg) {
                            if (0 != error) Toast.makeText(getApplicationContext(),"设置位置失败
" + errorMsg,Toast.LENGTH_LONG).show();
                        }
                    });
                } else {
                    tvLBS.setText(getString(R.string.text_live_lbs_fail));
                }
            } else {
                TCUserInfoMgr.getInstance().setLocation("", 0, 0, new ITCUserInfoMgrListener() {
                    @Override
                    public void OnQueryUserInfo(int error, String errorMsg) {
                    }
                    @Override
                    public void OnSetUserInfo(int error, String errorMsg) {
                        if (0 != error) Toast.makeText(getApplicationContext(),"设置位置失败" +
errorMsg,Toast.LENGTH_LONG).show();
                    }
                });
            }
        }
```

非卖品，仅供非商业用途或交流学习使用

设置选择摄像头直播还是录屏直播的代码如下。

```
    case R.id.btn_publish:
                    //trim 避免空格字符串
                    if (TextUtils.isEmpty(tvTitle.getText().toString().trim())) {
                        Toast.makeText(getApplicationContext(), "请输入非空直播标题",
 Toast.LENGTH_SHORT).show();
                    } else if(TCUtils.getCharacterNum(tvTitle.getText().toString())>TCConstants.
TV_TITLE_MAX_LEN){
                        Toast.makeText(getApplicationContext(), "直播标题过长 ,最大长度
为"+TCConstants.TV_TITLE_MAX_LEN/2, Toast.LENGTH_SHORT).show();
                    } else if (mUploading) {
                        Toast.makeText(getApplicationContext(), getString(R.string.publish_
 wait_uploading), Toast.LENGTH_SHORT).show();
                    } else if(!TCUtils.isNetworkAvailable(this)) {
                        Toast.makeText(getApplicationContext(), "当前网络环境不能发布直
播", Toast.LENGTH_SHORT).show();
                    } else {

                        if (mRecordType == TCConstants.RECORD_TYPE_SCREEN) {
                            //录屏
                            Intent intent = new Intent(this, TCScreenRecordActivity.class);
                            intent.putExtra(TCConstants.ROOM_TITLE,
                                    TextUtils.isEmpty(tvTitle.getText().toString()) ? TCUserInfoMgr.
 getInstance().getNickname() : tvTitle.getText().toString());
                            intent.putExtra(TCConstants.USER_ID, TCUserInfoMgr.getInstance().
 getUserId());
                            intent.putExtra(TCConstants.USER_NICK, TCUserInfoMgr.
 getInstance().getNickname());
                            intent.putExtra(TCConstants.USER_HEADPIC, TCUserInfoMgr.
 getInstance().getHeadPic());
                            intent.putExtra(TCConstants.COVER_PIC, TCUserInfoMgr.
 getInstance().getCoverPic());
                        //    intent.putExtra(TCConstants.SCR_ORIENTATION,
 mOrientation);
                            intent.putExtra(TCConstants.BITRATE, mBitrateType);
                            intent.putExtra(TCConstants.USER_LOC,
                                    tvLBS.getText().toString().equals(getString(R.string.
 text_live_lbs_fail)) ||
                                            tvLBS.getText().toString().equals(getString
(R.string.text_live_location)) ?
```

非卖品，仅供非商业用途或交流学习使用


```
                                                    getString(R.string.text_live_close_lbs) : tvLBS.
getText().toString());
                                startActivity(intent);
                                finish();
                            } else {
                                //摄像头
                                Intent intent = new Intent(this, TCLivePublisherActivity.class);
                                intent.putExtra(TCConstants.ROOM_TITLE,
                                        TextUtils.isEmpty(tvTitle.getText().toString()) ?
TCUserInfoMgr.getInstance().getNickname() : tvTitle.getText().toString());
                                intent.putExtra(TCConstants.USER_ID, TCUserInfoMgr.getInstance().
getUserId());
                                intent.putExtra(TCConstants.USER_NICK,
TCUserInfoMgr.getInstance().getNickname());
                                intent.putExtra(TCConstants.USER_HEADPIC, TCUserInfoMgr.
getInstance().getHeadPic());
                                intent.putExtra(TCConstants.COVER_PIC, TCUserInfoMgr.
getInstance().getCoverPic());
                                intent.putExtra(TCConstants.USER_LOC,
                                        tvLBS.getText().toString().equals(getString(R.string.
text_live_lbs_fail)) ||
                                                tvLBS.getText().toString().equals(getString
(R.string.text_live_location)) ?
                                                getString(R.string.text_live_close_lbs) : tvLBS.
getText().toString());
                                startActivity(intent);
                                finish();
                            }
                        }
                        break;
```

如果是录屏直播，选择输出的码率，代码如下。

```
mRGBitrate.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        switch (checkedId) {
            case R.id.rb_bitrate_slow:
                mBitrateType = TCConstants.BITRATE_SLOW;
                break;
            case R.id.rb_bitrate_normal:
```

非卖品，仅供非商业用途或交流学习使用


```
                    mBitrateType = TCConstants.BITRATE_NORMAL;
                    break;
                case R.id.rb_bitrate_fast:
                    mBitrateType = TCConstants.BITRATE_FAST;
                    break;
                default:
                    break;
            }
        }
    });
```

③ 在 TCPusherMgr.java 类中加入如下代码来实现与服务器进行推流控制开始直播与结束直播。

创建直播的代码如下。

```
    public void createLive(final String groupId, String title, String userId) {
        String url = "live/create";
        Map<String, String> object = new HashMap<>();
        object.put("groupId", groupId);
        object.put("title", title);
        object.put("userId", userId);
        object.put("coverPic", TCUserInfoMgr.getInstance().getCoverPic());
        TCHttpEngine.getInstance().postByForm(url, object, new TCHttpEngine.Listener() {
            @Override
            public void onResponse(int retCode, String retMsg, JSONObject retData) {
                if (null != retData) {
                    if (null != mPusherListener)
                        mPusherListener.onGetPusherUrl(retCode, groupId, TCUtils.JOToModel
(retData, ProgramInfo.class));
                } else {
                    if (null != mPusherListener)
                        mPusherListener.onGetPusherUrl(-1, groupId, null);
                }
            }
        });
    }
```

开始直播的代码如下。

```
    public void startLive(String streamId) {
        String url = "live/push/start";
        Map<String, String> params = new HashMap<>();
        params.put("streamId", streamId);
```

非卖品，仅供非商业用途或交流学习使用


```
        TCHttpEngine.getInstance().postByForm(url, params, new TCHttpEngine.Listener() {
            @Override
            public void onResponse(int retCode, String retMsg, JSONObject retData) {
            }
        });
    }
```

结束直播的代码如下。

```
    public void endLive(String streamId) {
        String url = "live/push/stop";
        Map<String, String> params = new HashMap<>();
        params.put("streamId", streamId);
        TCHttpEngine.getInstance().postByForm(url, params, new TCHttpEngine.Listener() {
            @Override
            public void onResponse(int retCode, String retMsg, JSONObject retData) {

            }
        });
    }
```

要创建摄像头直播，在 TCLivePublisherActivity.java 类中的 onJoinGroupCallback 方法中调用 TCPusherMgr 中的 createLive 方法，代码如下。

```
    @Override
    public void onJoinGroupCallback(int code, String msg) {
        if(0 == code) {
            //获取推流地址
            Log.d(TAG, "onJoin group success" + msg);
            mTCPusherMgr.createLive(msg, mTitle, TCUserInfoMgr.getInstance().getGUserId());
        } else if (TCConstants.NO_LOGIN_CACHE == code) {
            TXLog.d(TAG, "onJoin group failed" + msg);
            showErrorAndQuit(TCConstants.ERROR_MSG_NO_LOGIN_CACHE);
        } else {
            TXLog.d(TAG, "onJoin group failed" + msg);
            showErrorAndQuit(TCConstants.ERROR_MSG_JOIN_GROUP_FAILED + code);
        }
    }
```

开始直播推流的代码如下。

```
    private void startPublish() {
        if (mTXLivePusher == null) {
            mTXLivePusher = new TXLivePusher(this);
            mTXLivePusher.setPushListener(this);
```

```
            mTXPushConfig.setAutoAdjustBitrate(false);
            mTXPushConfig.setVideoResolution(TXLiveConstants.VIDEO_RESOLUTION_
TYPE_540_960);
            mTXPushConfig.setVideoBitrate(1000);
            if (TCHWSupportList.isHWVideoEncodeSupport()){
                mTXPushConfig.setHardwareAcceleration(true);
            }
            else{
                Log.d(TAG,"当前机型不支持视频硬编码");
                mTXPushConfig.setVideoResolution(TXLiveConstants.VIDEO_RESOLUTION
_TYPE_360_640);
            }
            //切后台推流图片
            BitmapFactory.Options options = new BitmapFactory.Options();
            options.inPreferredConfig = Bitmap.Config.ARGB_8888;
            Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.pause_
publish,options);
            mTXPushConfig.setPauseImg(bitmap);
            mTXLivePusher.setConfig(mTXPushConfig);
        }
        mAudioCtrl.setPusher(mTXLivePusher);
        mTXCloudVideoView.setVisibility(View.VISIBLE);
            //mBeautySeekBar.setProgress(100);

        mTXLivePusher.startCameraPreview(mTXCloudVideoView);
        mTXLivePusher.startPusher(mPushUrl);
    }
```

结束直播推流的代码如下。

```
    private void stopPublish() {
        if (mTXLivePusher != null) {
            mTXLivePusher.stopCameraPreview(false);
            mTXLivePusher.setPushListener(null);
            mTXLivePusher.stopPusher();
        }
        if (mAudioPlayer != null) {
            mAudioPlayer.stop();
            mAudioPlayer = null;
        }
        if(mAudioCtrl != null){
            mAudioCtrl.unInit();
```

```
            mAudioCtrl = null;
        }
    }
```

④ 在 TCLivePublisherActivity.java 类中的 onGetPusherUrl 方法中加入如下代码来实现开始直播推流。

```
    @Override
    public void onGetPusherUrl(int errCode, String groupId, ProgramInfo programInfo) {
        if (programInfo != null) {
            mProgram = programInfo;
            mPushUrl = programInfo.getPushUrl();
            startRecordAnimation();
            //开启推流
            startPublish();
        } else {
            if (null == groupId) {
                showErrorAndQuit(TCConstants.ERROR_MSG_CREATE_GROUP_FAILED +
errCode);
            } else {
                showErrorAndQuit(TCConstants.ERROR_MSG_GET_PUSH_URL_FAILED +
errCode);
            }
        }
    }
```

退出房间，同时调用 TCPusherMgr 中的 endLive 方法结束直播，代码如下。

```
    public void quitRoom() {
        mTCChatRoomMgr.deleteGroup();
        TCPusherMgr.getInstance().endLive(mProgram.getStreamId());
    }
```

获取观看详细信息的代码如下。

```
    public void showDetailDialog() {
        //确认则显示观看 detail
        stopRecordAnimation();
        DetailDialogFragment dialogFragment = new DetailDialogFragment();
        Bundle args = new Bundle();
        args.putString("time", TCUtils.formattedTime(mSecond));
        args.putString("heartCount", String.format(Locale.CHINA, "%d", lHeartCount));
        args.putString("totalMemberCount", String.format(Locale.CHINA, "%d", lTotalMemberCount));
        dialogFragment.setArguments(args);
```

```
        dialogFragment.setCancelable(false);
        dialogFragment.show(getFragmentManager(), "");
    }
```

在开始直播的同时开启计时红点，代码如下。

```
    private void startRecordAnimation() {
        mObjAnim = ObjectAnimator.ofFloat(mRecordBall, "alpha", 1f, 0f, 1f);
        mObjAnim.setDuration(1000);
        mObjAnim.setRepeatCount(-1);
        mObjAnim.start();

        //直播时间
        if(mBroadcastTimer == null) {
            mBroadcastTimer = new Timer(true);
            mBroadcastTimerTask = new BroadcastTimerTask();
            mBroadcastTimer.schedule(mBroadcastTimerTask, 1000, 1000);
        }
    }
```

在直播结束时计时红点停止，代码如下。

```
    private void stopRecordAnimation() {
        if (null != mObjAnim)
            mObjAnim.cancel();
        //直播时间
        if (null != mBroadcastTimer) {
            mBroadcastTimerTask.cancel();
        }
    }
```

⑤ 在 TCLivePublisherActivity.java 类中的 onPushEvent 方法中调用 TCPusherMgr 中的 startLive 与 endLive 方法来开始直播和结束直播，代码如下。

```
    @Override
      public void onPushEvent(int event, Bundle bundle) {
          if (event < 0) {
              if (event == TXLiveConstants.PUSH_ERR_NET_DISCONNECT) {//网络断开，
弹出对话框强提醒，推流过程中直播中断需要显示直播信息后退出
                  showComfirmDialog(TCConstants.ERROR_MSG_NET_DISCONNECTED, true);
              } else if (event == TXLiveConstants.PUSH_ERR_OPEN_CAMERA_FAIL) {//未
获得摄像头权限，弹出对话框强提醒，并退出
                  showErrorAndQuit(TCConstants.ERROR_MSG_OPEN_CAMERA_FAIL);
              } else if (event == TXLiveConstants.PUSH_ERR_OPEN_MIC_FAIL) { //未获得
```

```
麦克风权限，弹出对话框强提醒，并退出
                    Toast.makeText(getApplicationContext(), bundle.getString (TXLiveConstants.
EVT_DESCRIPTION), Toast.LENGTH_SHORT).show();
                    showErrorAndQuit(TCConstants.ERROR_MSG_OPEN_MIC_FAIL);
                } else {
                    //其他错误弹出 Toast 弱提醒，并退出
                    Toast.makeText(getApplicationContext(), bundle.getString(TXLiveConstants.
EVT_DESCRIPTION), Toast.LENGTH_SHORT).show();

                    mTXCloudVideoView.onPause();
                    // 异常
                    TCPusherMgr.getInstance().endLive(mProgram.getStreamId());
                    stopRecordAnimation();
                    finish();
                }
            }
        if (event == TXLiveConstants.PUSH_EVT_PUSH_BEGIN) {
                // 直播推流开始
                TCPusherMgr.getInstance().startLive(mProgram.getStreamId());
            }
        }
```

⑥ 停止并初始化直播程序，代码如下。

```
@Override
protected void onDestroy() {
    super.onDestroy();
    stopRecordAnimation();
    mTXCloudVideoView.onDestroy();
    stopPublish();
    TXRtmpApi.setRtmpDataListener(null);
    mTCChatRoomMgr.removeMsgListener();
    mTCPusherMgr.setPusherListener(null);
}
```

注意

在所有退出当前界面的操作中都调用 stoppusher()，录屏直播请参考摄像头直播尝试编写。

（2）执行测试

① 执行效果。

单击快捷工具栏中的“运行”按钮▶，运行程序。使用用户名“douxue1”、密码“123456789”登录，单击“直播”按钮设置，完成后开始直播，如图 5-3 所示。

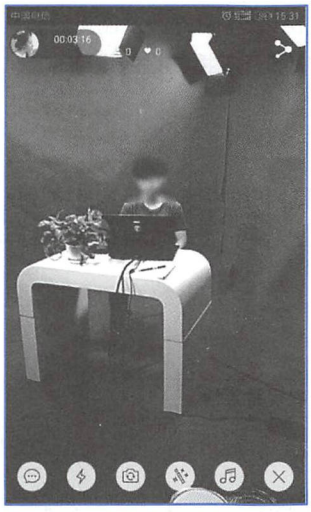


图 5-3
执行启动后的效果图

② 手动测试。

测试的用例（TestCase）如表 5-1 所示，可以编写单元测试或者自己手动测试。

测试场景	输入参数	预期结果
没有输入标题名次		提示请设置标题
设置完全正确的		开启直播

表 5-1
测试用例

经过测试，直播设置界面设置完成后可以正常开启直播。至此，本任务已经实现了开启直播功能的开发。

项目实训

实训题目

查询与直播推流相关的知识，编写直播推流技术调研报告。

实训目的

1. 了解直播推流的原理。
2. 了解直播推流的应用场景。
3. 了解直播推流开启与结束的方法。

实训内容

从网络搜索相关直播推流知识，了解直播推流的技术发展状况和应用领域等。

任务 5.2　实现直播列表模块

任务描述

在 Web 后台实现一个能够取得当前正在直播的房间和所有直播结束的房间功能接

口，提供给 Android 程序调用。APP 客户端通过调用后台提供的接口能够展示出一个直播列表，列表中包含正在直播的房间及直播结束的房间。

知识学习

直播后台的核心职能是管理和维护直播间列表的数据，并对外提供针对直播间列表进行增、删、改、查的业务接口，一个视频直播解决方案的大部分功能都是围绕直播间在做文章，如图 5-4 所示。

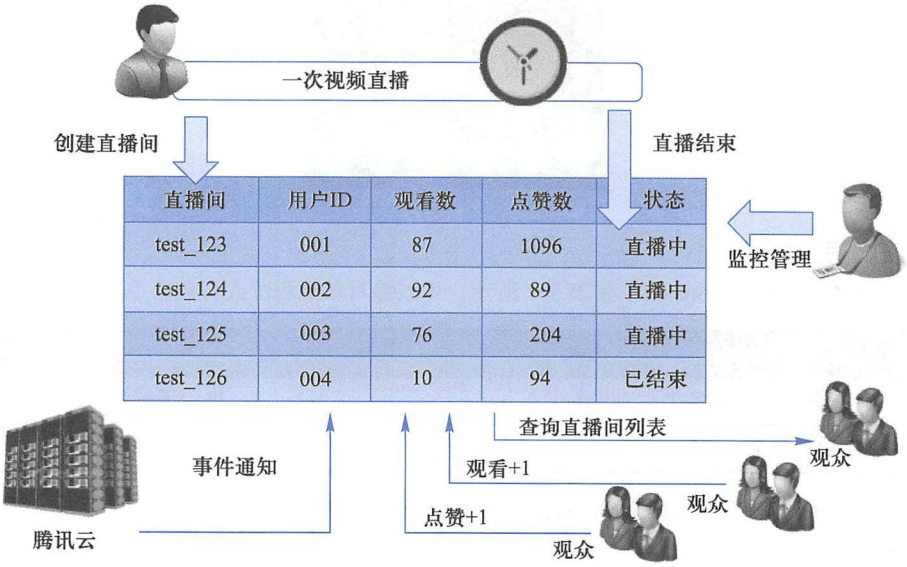


图 5-4
直播后台职能

1. 创建直播间（增）

当一个主播开始直播时，需要先创建一个直播间，这就等于是在直播间列表中增加一条新的数据。

2. 关闭直播间（删）

当一个直播结束后，APP 有必要通知后台把刚才的直播间状态置为直播结束，或者干脆将其从列表中删除。

3. 修改直播间状态（改）

- 当有新的观众加入时，意味着某个直播间的观众数要+1。
- 当有观众给主播点赞时，意味着某个直播间的点赞数要+1。
- 当有一条视频流意外断流时，会收到来自腾讯云的通知，意味着某个直播间的状态要进行修正。
- 当监管人员发现某一直播间内容涉及违规行为时，需要对其禁播，意味着某个直播间的状态要改为已禁播。

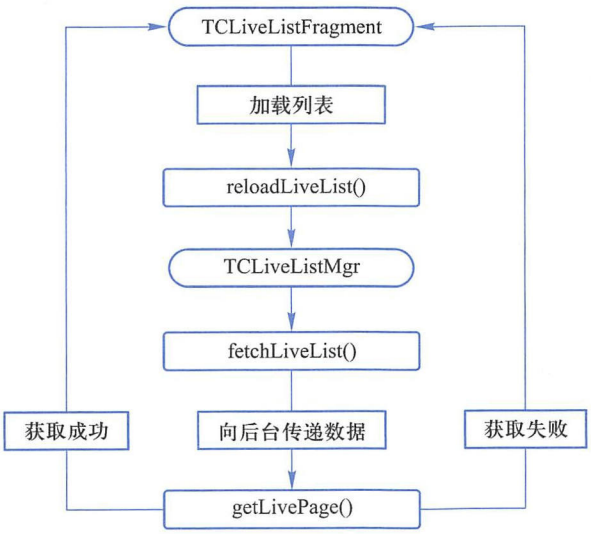
4. 查询直播间列表（查）

每一个打开 APP 的观众，都会到直播后台查询一下当前的直播间列表。同时，相关

运营人员也需要实时查看直播间的状态，以便进行必要的监管。

任务实现

直播列表功能具体流程如图 5-5 所示。



微课 5-2
实现直播列表模块

图 5-5
直播列表流程图

1. 后台直播列表接口开发

（1）直播列表

通过 Web 的后台接口，将之前直播过或者正在直播的用户显示在用户手机 APP 之中。

（2）实现步骤

首先，APP 客户端发送一个获取分页直播列表的请求到后台，并附带 pageNo、pageSize 和 live 这 3 个参数。pageNo 表示当前是第几页；pageSize 表示每页应该显示多少条数据；live 表示搜索的条件，可以设定根据直播房间的名称、streamId、直播房间状态等来筛选直播列表数据。

源码 5-2:
实现直播列表模块

① 打开 Eclipse，导入本教材提供的配套项目包。

② 在 com.douxue.live.controller.app.live 中的 LiveController.java 文件中添加代码，如下所示。

```
/**
 * 获取直播分页列表
 *
 * @param pageNo 当前页
 * @param pageSize 每页条数
 * @param live 查询条件
 * @return
 */
@RequestMapping("getLivePage")
```

```
    @ResponseBody
    public RestResult getLivePage(@RequestParam(value = "pageNo", defaultValue = "1")
Integer pageNo, @RequestParam(value = "pageSize", defaultValue = "5") Integer pageSize, Live
live) {
        PageData<Live> page = liveService.getAppLivePage(pageNo, pageSize, live);
        return new RestResult(page);
    }
```

③ 在 com.douxue.live.service.live 中的 LiveService.java 文件中添加以下代码。

```
/**
 * 获取 APP 直播分页列表
 *
 * @param pageNum
 * @param pageSize
 * @param live
 * @return
 *
 */
public PageData<Live> getAppLivePage(Integer pageNum, Integer pageSize, Live live);
```

④ 在 com.douxue.live.service.live.impl 中的 LiveServiceImpl.java 文件夹中实现接口获取 APP 直播分页列表的具体方法，代码如下，pageNum 为页数，pageSize 为一页内直播间数量。

```
@Autowired
private LiveRedisDao liveRedisDao;

@Override
public PageData<Live> getAppLivePage(Integer pageNum, Integer pageSize, Live live) {
    PageParam<Live> param = new PageParam<Live>(live, pageNum, pageSize);
    List<Live> data = liveMapper.getAppLivePage(param);
    if (null != data && data.size() > 0) {
        for (Live l : data) {
            l = setLiveRedisData(l);
        }
    }
    PageData<Live> page = new PageData<Live>(pageNum, pageSize, param.getDataTotal(), data);
    return page;
}
private Live setLiveRedisData(Live live) {
    if (null == live) {
```


版权所有，严禁传播，违者自负法律责任！

```
            return null;
        }
        // 观看次数
    live.setViewerCount(liveRedisDao.getLiveViewerCount(live.getStreamId()));
        // 点赞
    live.setPraiseCount(liveRedisDao.getPraiseCount(live.getStreamId()));
        return live;
    }
```

⑤ 在 com.douxue.live.dao.mapper 中实现 LiveMapper.java，代码如下。

```
List<Live> getAppLivePage(PageParam<Live> param);
```

⑥ 在 LiveMapper.xml 中实现分页查询的具体方法，代码如下。

```
    <select id="getAppLivePage"resultMap="BaseResultMap"parameterType="com.douxue.live.
common.page.PageParam">
    SELECT
    <include refid="Base_Column_List" />
    FROM t_live
    WHERE state IN (1,100)
        <if test="p.userId != null and p.userId != ''">
        AND userId = #{p.userId}
      </if>
      <if test="p.streamId != null and p.streamId != ''">
        AND stream_id = #{streamId}
      </if>
      <if test="p.title != null and p.title != ''">
        AND title LIKE CONCAT('%',#{p.title},'%')
      </if>
      <if test="p.state != null">
        AND state = #{p.state}
      </if>
    ORDER BY create_time DESC
    </select>
```

⑦ 在 com.douxue.live.dao.redis 中的 LiveRedisDao.java 文件中添加以下代码，用于获取直播间内的观众人数、点赞数的数据。

```
    @Autowired
    private RedisHashService redisHashService;

    // 直播 Redis KEY 前缀
```


版权所有，严禁传播，违者自负法律责任！

```
    private static final String LIVE_KEY_PREFIX = "LIVE_";
    // 点赞域 KEY
    private static final String LIVE_PRAISE_COUNT_FIELD = "PRAISE_COUNT";
    // 播放次数域 KEY
    private static final String LIVE_VIEWER_COUNT_FIELD = "VIEWER_COUNT";

    private String getLiveKey(String streamId) {
        return new StringBuilder().append(LIVE_KEY_PREFIX).append(streamId).toString();
    }
    /**
     * 获取观众数
     *
     * @param streamId
     * @return
     *
     */
    public long getLiveViewerCount(String streamId) {
        String key = getLiveKey(streamId);
        Long viewerCnt = redisHashService.getHashValue(key, LIVE_VIEWER_COUNT_FIELD,
Long.class);
        return viewerCnt != null ? viewerCnt : 0;
    }

    /**
     * 获取点赞数
     *
     * @param streamId
     * @return
     *
     */
    public long getPraiseCount(String streamId) {
        String key = getLiveKey(streamId);
        Long praiseCnt = redisHashService.getHashValue(key, LIVE_PRAISE_COUNT_FIELD,
Long.class);
        return praiseCnt != null ? praiseCnt : 0;
```

2. Android 直播列表功能开发

(1) 实现步骤

① 打开 Android Studio，导入本教材提供的配套项目包。

版权所有，严禁传播，违者自负法律责任！

② 创建一个名为 TCLiveListAdapter.java 的类，此类为 listview 适配器，代码如下。

```
public class TCLiveListAdapter extends ArrayAdapter<ProgramInfo> {
    private int resourceId;
    private Activity mActivity;

    private class ViewHolder {
        TextView tvTitle;
        TextView tvHost;
        TextView tvMembers;
        TextView tvAdmires;
        TextView tvLbs;
        ImageView ivCover;
        ImageView ivAvatar;
        ImageView ivLogo;
    }

    public TCLiveListAdapter(Activity activity, ArrayList<ProgramInfo> objects) {
        super(activity, R.layout.listview_video_item, objects);
        resourceId = R.layout.listview_video_item;
        mActivity = activity;

    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder;

        if (convertView != null) {
            holder = (ViewHolder) convertView.getTag();
        } else {
            convertView = LayoutInflater.from(getContext()).inflate(resourceId, null);
            holder = new ViewHolder();
            holder.ivCover = (ImageView) convertView.findViewById(R.id.cover);
            holder.tvTitle = (TextView) convertView.findViewById(R.id.live_title);
            holder.tvHost = (TextView) convertView.findViewById(R.id.host_name);
            holder.tvMembers = (TextView) convertView.findViewById(R.id.live_members);
            holder.tvAdmires = (TextView) convertView.findViewById(R.id.praises);
            holder.tvLbs = (TextView) convertView.findViewById(R.id.live_lbs);
            holder.ivAvatar = (ImageView) convertView.findViewById(R.id.avatar);
```


版权所有，严禁传播，违者自负法律责任！

```
        holder.ivLogo = (ImageView) convertView.findViewById(R.id.live_logo);

        convertView.setTag(holder);
    }

    ProgramInfo data = getItem(position);

    //直播封面
    String cover = data.getCoverPic();
    if (TextUtils.isEmpty(cover)) {
        holder.ivCover.setImageResource(R.drawable.bg);
    } else {
        RequestManager req = Glide.with(mActivity);
        req.load(cover).placeholder(R.drawable.bg).into(holder.ivCover);
    }

    //主播头像
    TCUtils.showPicWithUrl(mActivity, holder.ivAvatar, "url", R.drawable.face);
    //主播昵称
    if (TextUtils.isEmpty(data.getUser().getNickName())) {
        holder.tvHost.setText("@" + TCUtils.getLimitString(data.getUserId(), 10));
    } else {
        holder.tvHost.setText("@"+TCUtils.getLimitString(data.getUser().getNickName(),10));
    }
    //主播地址
    if (TextUtils.isEmpty(data.getLocation())) {
        holder.tvLbs.setText(getContext().getString(R.string.live_unknown));
    } else {
        holder.tvLbs.setText(TCUtils.getLimitString(data.getLocation(), 9));
    }

    //直播标题
    holder.tvTitle.setText(TCUtils.getLimitString(data.getTitle(), 10));
    //直播观看人数
    holder.tvMembers.setText("" + data.getViewerCount());
    //直播点赞数
    holder.tvAdmires.setText("" + data.getPraiseCount());
    //视频类型，直播或者回放
    if (data.getState()== 1) {
        holder.ivLogo.setImageResource(R.drawable.live);
    } else {
```


版权所有，严禁传播，违者自负法律责任！

```
                holder.ivLogo.setImageResource(R.drawable.playback);
            }
            return convertView;
        }

    }
```

③ 在 TCLiveListMgr.java 类中加入如下代码来获取服务器中的数据。

```
    public boolean reloadLiveList(Listener listener) {
        if (mIsFetching) {
            TXLog.w(TAG, "reloadLiveList ignore when fetching");
            return false;
        }
        TXLog.d(TAG, "fetchLiveList start");
        mLiveInfoList.clear();
//          fetchLiveList(LIST_TYPE_ALL, 1, PAGESIZE, listener);
        fetchLiveList(listener);
        mIsFetching = true;
        return true;
    }
```

④ 在 TCLiveListFragment.java 类中获取服务器中的数据，并绑定适配器填充适配器中的集合。

```
mVideoListViewAdapter = new TCLiveListAdapter(getActivity(),
        (ArrayList<ProgramInfo>) TCLiveListMgr.getInstance().getLiveList().clone());
mVideoListView.setAdapter(mVideoListViewAdapter);
```

⑤ 加入如下方法，调用 TCLiveListMgr.java 类中的 reloadLiveList 方法来获取集合。

```
private boolean reloadLiveList() {
    return TCLiveListMgr.getInstance().reloadLiveList(new TCLiveListMgr.Listener() {
        @Override
        public void onLiveList(int retCode,final ArrayList<ProgramInfo>result, boolean refresh) {
            if (null != result && !result.isEmpty()) {
                mVideoListViewAdapter.clear();
                if (result != null) {
                mVideoListViewAdapter.addAll((ArrayList<ProgramInfo>) result.clone());
                }
                if (refresh) {
                    mVideoListViewAdapter.notifyDataSetChanged();
                }
```


版权所有，严禁传播，违者自负法律责任！

```
                } else {
                    Toast.makeText(getActivity(), "刷新列表失败", Toast.LENGTH_LONG).show();
                }
                mSwipeRefreshLayout.setRefreshing(false);
            }
        });
    }
```

（2）执行测试

① 执行效果。

单击快捷工具栏中的“运行”按钮▶，运行程序。使用用户名“douxue1”、密码“123456789”登录，如图 5-6 所示。

图 5-6
执行启动后的效果图

② 手动测试。

测试的用例（TestCase）如表 5-2 所示，可以编写单元测试或者自己手动测试。

表 5-2
测试用例

测试场景	输入参数	预期结果
正确登录	—	展示直播列表

经过测试，直播设置界面设置完成后可以正常开启直播。至此，本任务已经实现了开启直播功能的开发。

项目实训

实训题目

练习 ListView 的应用。

实训目的

1. 了解 ListView 的原理。

版权所有，严禁传播，违者自负法律责任！

2. 了解 Adapter 的使用。

实训内容

在 Android Studio 中实现一个静态列表。

任务 5.3　实现观看直播模块

任务描述

在 Web 后台实现一个能够根据直播 ID 获取直播流的地址，然后将这个地址返回给 APP 客户端，APP 客户端可以根据这个直播流的地址拉取直播流，实现直播视频的播放。

知识学习

RTMP SDK 包含推流和播放两方面功能，推流为主播端功能，播放（分为直播和点播）为观众端功能。

1. 直播和点播

直播的视频源是实时生成的，有人推流直播才有意义。所以，一旦主播停播，直播 URL 也就失效了，而且由于是实时直播，所以播放器在播直播视频的时候是没有进度条的。

点播的视频源是云端的一个文件，只要文件没有被提供方删除，就随时可以播放，而且由于整个视频都在服务器上，所以播放的时候是有进度条的。

2. 协议的支持

通常使用的直播协议如下，APP 端推荐使用 FLV 协议的直播地址（以 http 打头，以.flv 结尾），如图 5-7 所示。

直播协议	优点	缺点	播放延迟
FLV	成熟度高、高并发无压力	需集成SDK才能播放	2～3s
RTMP	优质线路下理论延迟最低	高并发情况下表现不佳	1～3s
HLS(m3u8)	手机浏览器支持度高	延迟非常高	10～30s

图 5-7
直播协议

任务实现

观看直播功能具体流程如图 5-8 所示。

版权所有，严禁传播，违者自负法律责任！

微课 5-3
实现观看直播模块

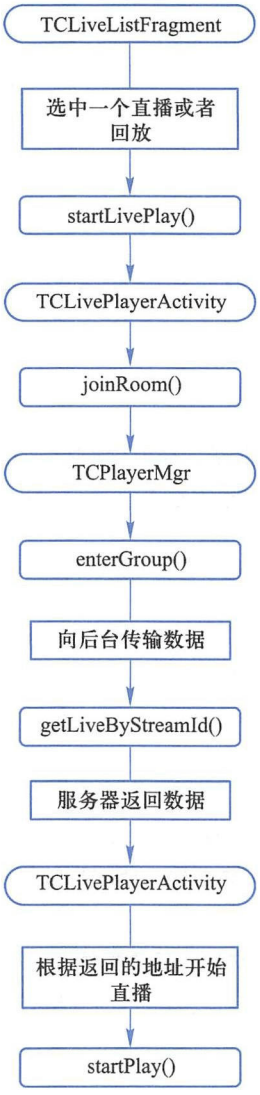


图 5-8
观看直播流程图

1. 后台观看直播接口开发

(1) 进入直播间

用户通过单击直播列表中的直播内容，可以在跳转的界面看到开直播的用户的直播内容。

(2) 实现步骤

首先，APP 客户端发送一个获取直播播放地址的请求到后台，并附带 streamId 参数，利用 streamId 获取直播的详细内容，其中包括直播的播放地址。

源码 5-3:
实现观看直播模块

① 打开 Eclipse，导入本教材提供的配套项目包。

② 在 com.douxue.live.controller.app.live 中的 LiveController.java 文件中添加获取直播等代码，如下所示。

```
/**
*获取直播
```

```
*
*@param streamId
* @return
*/
@RequestMapping("getLiveByStreamId")
@ResponseBody
public RestResult getLiveByStreamId(@RequestParam("streamId") String streamId) {
    Live live = liveService.getLiveByStreamId(streamId);
    return new RestResult(live);
}
```

③ 在 com.douxue.live.service.live 中的 LiveService.java 文件中添加获取直播的接口代码，如下所示。

```
/**
 * 根据 streamId 获取直播
 *
 * @param streamId
 * @return
 *
 */
public Live getLiveByStreamId(String streamId);
```

④ 在 com.douxue.live.service.live.impl 中的 LiveServiceImpl.java 类中实现 LiveService.java 中的接口方法，代码如下。

```
@Override
public Live getLiveByStreamId(String streamId) {
    Live live = liveMapper.selectByPrimaryKey(streamId);
    live = setLiveRedisData(live);
    return live;
}
```

2. Android 观看直播功能开发

(1) 实现步骤

① 打开 Android Studio，导入本教材提供的配套项目包。

② 在 TCPlayerMgr.java 文件中加入如下代码，来实现与服务器通信。

```
public void enterGroup(String userId, String liveUserId, String groupId,
                       String nickname, String headPic, int flag) {
    mPlayerListener.onRequestCallback(0);
    try {
        JSONObject req = new JSONObject();
```

```
            req.put("Action", "EnterGroup");
            req.put("userid", userId);
            req.put("liveuserid", liveUserId);
            req.put("groupid", groupId);
            req.put("nickname", nickname);
            req.put("headpic", headPic);
            req.put("flag", flag);

            TCHttpEngine.getInstance().post(req, new TCHttpEngine.Listener() {
                @Override
                public void onResponse(int retCode, String retMsg, JSONObject retData) {
                    TXLog.d(TAG, "EnterGroup: retCode --" + retCode + "|" + retMsg);
                    if (mPlayerListener != null) {
                        mPlayerListener.onRequestCallback(retCode);
                    }
                    mPlayerListener = null;
                }
            });

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

③ 在 TCLiveListFragment.java 文件中加入如下代码来跳转至观看直播界面。

```
private void startLivePlay(final ProgramInfo item) {
    Intent intent = new Intent(getActivity(), TCLivePlayerActivity.class);
    intent.putExtra(TCConstants.ROOM_ID,item.getUserId());
    intent.putExtra(TCConstants.PLAY_URL, item.getState()== 1 ? item.getPlayFlvUrl() :
            item.getLiveVideos().get(0).getVideoUrl());
    intent.putExtra(TCConstants.PUSHER_ID, item.getUser().getNickName()== null ?
            item.getUserId() : item.getUser().getNickName());
    intent.putExtra(TCConstants.PUSHER_AVATAR, "url");
    intent.putExtra(TCConstants.HEART_COUNT, "" + item.getPraiseCount());
    intent.putExtra(TCConstants.MEMBER_COUNT,"" + item.getViewerCount());
    intent.putExtra(TCConstants.GROUP_ID, item.getGroupId());
    intent.putExtra(TCConstants.PLAY_TYPE, item.getState()== 1);
    intent.putExtra(TCConstants.FILE_ID, "");
    intent.putExtra(TCConstants.COVER_PIC, item.getCoverPic());
    intent.putExtra(TCConstants.LIVE_OBJECT, item);
```

```
        startActivityForResult(intent,START_LIVE_PLAY);
    }
```

④ 在 TCLivePlayerActivity.java 文件中加入如下代码，来展示观众进入直播后展示的页面，并在初始化中调用。

```
    public void joinRoom() {
        //初始化消息回调，当前存在：获取文本消息、用户加入/退出消息、群组解散消息、点赞消息、弹幕消息回调
        mTCPlayerMgr = TCPlayerMgr.getInstance();
        mTCPlayerMgr.setPlayerListener(this);
        }
        mTCPlayerMgr.enterGroup(mUserId, mPusherId, mIsLivePlay ? mGroupId : mFileId, mNickname, mHeadPic, mIsLivePlay ? 0 : 1);
        startPlay();
    }
```

⑤ 开始直播画面，代码如下。

```
    private void startPlay() {
        if (mTXLivePlayer == null) {
            mTXLivePlayer = new TXLivePlayer(this);
        }
        mTXLivePlayer.setPlayerView(mTXCloudVideoView);
        mTXLivePlayer.setRenderRotation(TXLiveConstants.RENDER_ROTATION_PORTRAIT);
        mTXLivePlayer.setRenderMode(TXLiveConstants.RENDER_MODE_FULL_FILL_SCREEN);
        mTXLivePlayer.setPlayListener(this);
        mTXLivePlayer.setConfig(mTXPlayConfig);
        mTXLivePlayer.setLogLevel(TXLiveConstants.LOG_LEVEL_DEBUG);
        int result;
        if (mIsLivePlay) {
            result = mTXLivePlayer.startPlay(mPlayUrl, TXLivePlayer.PLAY_TYPE_LIVE_FLV);
        } else {
            result = mTXLivePlayer.startPlay(mPlayUrl, TXLivePlayer.PLAY_TYPE_VOD_MP4);
        }
        if (0 != result) {
            Log.e("djl", "result != 0 ,mPlayUrl:" + mPlayUrl);
            Intent rstData = new Intent();

            if (-1 == result) {
                Log.d(TAG, TCConstants.ERROR_MSG_NOT_QCLOUD_LINK);
                rstData.putExtra(TCConstants.ACTIVITY_RESULT, TCConstants.ERROR_MSG_NOT_QCLOUD_LINK);
```

```
                } else {
                    Log.d(TAG, TCConstants.ERROR_RTMP_PLAY_FAILED);
                    rstData.putExtra(TCConstants.ACTIVITY_RESULT, TCConstants.ERROR_MSG_
NOT_QCLOUD_LINK);
                }
                mTXCloudVideoView.onPause();
                stopPlay(true);
                setResult(TCLiveListFragment.START_LIVE_PLAY, rstData);
                finish();
            } else {
                mPlaying = true;
                Log.e("djl", "result == 0 ,mPlayUrl:" + mPlayUrl);
            }
        }
```

（2）执行测试

① 执行效果。

单击快捷工具栏中的“运行”按钮▶，运行程序。使用用户名“douxue1”、密码“123456789”登录，单击第一个直播房间。如图 5-9 所示。

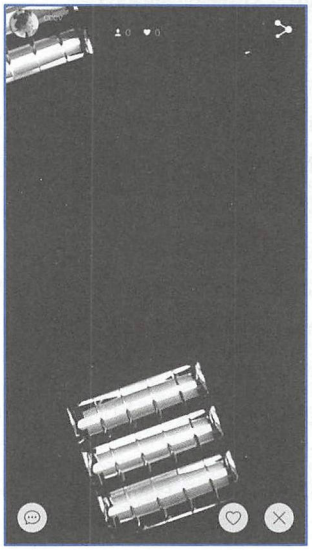


图 5-9
执行启动后的效果图

② 手动测试。

测试的用例（TestCase）如表 5-3 所示，可以编写单元测试或者自己手动测试。

表 5-3
测试用例

测试场景	输入参数	预期结果
正确登录，单击直播	—	开始播放

经过测试，可以观看直播。至此，本任务已经实现了观看功能的开发。

项目实训

实训题目

查询与前后台通信相关的知识，编写前后台通信技术调研报告。

实训目的

1. 了解前后台通信的原理。
2. 了解接口的实现方法。

实训内容

从网络搜索相关前后台通信的知识，了解前后台通信有哪些技术并撰写报告等。

单元小结

本单元介绍了整个直播的基本流程，包括主播开启直播、直播列表刷新、用户单击列表中的数据进入直播间观看直播。本单元重点阐述了如何进行前后台通信，同时本单元中涉及了简单的推流服务及 ListView 控件的应用，相信读者在学习之后对直播的流程会有进一步的理解。

单元 6 直播互动模块

学习目标

PPT6:
直播互动模块

知识目标

- 了解 Android 响应事件设置的方法。
- 了解实现弹幕的原理。
- 了解调用系统摄像头的方法。

技能目标

- 掌握点赞功能的实现方法。
- 掌握直播弹幕功能的实现方法。
- 掌握拓展功能的实现方法。

学习情境

斗学直播 APP 在小赵的努力下已经完成了 APP 的登录注册等功能，用户可以登录系统查看和修改自己的个人信息，也可以进行直播，但是不能进行直播互动，同时直播间界面效果比较差。小赵计划接下来完善直播 APP 的直播互动模块，要求直播 APP 能让主播与观众可以通过弹幕进行交流，观众可以为主播点赞。

任务 6.1　实现点赞模块

任务描述

了解 Android 几种响应事件的实现方式，在观看直播界面根据所学响应事件完成对心形按钮的响应事件设置。

知识学习

在 Android 的开发中，对于点击事件的 OnClickListener 有 4 种实现方式，可以根据实际场景的需要选择合适的用法。下面以 Button 按钮来举例说明。

（1）方法 1

适合场景：任何场景都通用，但对于一个 Activity 中有多个控件要实现 onClick()方法就会显得代码冗余。

```
Button bt_Demo = (Button)findViewById(R.id.bt_Demo);
bt_Demo.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
//具体单击操作的逻辑
}
});
```

（2）方法 2

适合场景：适合有多个同类型控件（如 Button 数组）要实现 onClick()方法时使用。

```
Button[] demoBtns ;
……
for(Button button : demoBtns ){
button.setOnClickListener(listener);
}
private OnClickListener listener = new OnClickListener(){
@Override
```

```
public void onClick(View arg0) {
// TODO Auto-generated method stub
switch(arg0.getId()){
case R.id.btn_Demo:
//具体点击操作的逻辑
break;
default:
break;
}
}
}
```

（3）方法 3

适合场景：适合有多个同类型控件（如 Button 数组）要实现 onClick()方法时使用。

```
Button bt_Demo = (Button)findViewById(R.id.bt_Demo);
bt_Demo.setOnClickListener(new ButtonListener());
private class ButtonListener implements OnClickListener{
@Override
public void onClick(View arg0) {
// TODO Auto-generated method stub
switch(arg0.getId()){
case R.id.btn_Demo:
//具体点击操作的逻辑
break;
default:
break;
}
}
}
```

（4）方法 4

适合场景：适合界面上有不同类型的控件，这种方式将所有控件的 onClick()方法在一个方法里面实现，看起来比较简洁。

在 Activity 中实现 OnClickListener 接口的代码如下。

```
public class MyActivity extends Activity implements OnClickListener {
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
```

```
//按钮
Button btn_Demo = (Button)findViewById(R.id.bt_Demo);
bt_Demo.setOnClickListener(this);
}
@Override
public void onClick(View v) {
switch(arg0.getId()){
case R.id.btn_Demo:
//具体点击操作的逻辑
break;
default:
break;
}
}
}
```

（5）方法 5

适合场景：任何场景都通用，特别是某个控件的单击方法要实现的逻辑较复杂的时候。

在布局文件中加上 android:onClick="方法名"，对于有些控件（如 TextView）需要兼容旧的 SDK API，还需要加上 android:clickable="true"。

在 XML 文件中的代码如下。

```
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
android:onClick="onTextViewClick"
android:text="点击事件"
android:textSize="16sp" />
```

在 Java 文件中的代码如下。

```
//在代码中实现 onTextViewClick()方法即可。
public void onTextViewClick(){
//具体点击操作的逻辑
}
```

微课 6-1
实现点赞模块

任务实现

点赞功能具体流程如图 6-1 所示。

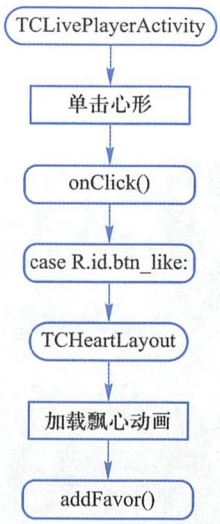


图 6-1
点赞流程图

（1）实现步骤

源码 6-1：
实现点赞模块

① 打开 Android Studio，导入本教材提供的配套项目包。

② 在 TCHeartLayout.java 文件中加入如下方法，并在 TCLivePlayerActivity.java 中的心形控件的监听事件中调用此方法，来实现点赞飘心动画功能。

```
public void addFavor() {
    TCHeartView heartView = new TCHeartView(getContext());
    heartView.setDrawable(mHeartsDrawable[mRandom.nextInt(8)]);
    mAnimator.start(heartView, this);
}
```

③ 心形监听事件代码如下。

```
case R.id.btn_like:
    if (mHeartLayout != null) {
        mHeartLayout.addFavor();
    }

    //点赞发送请求限制
    if (mLikeFrequeControl == null) {
        mLikeFrequeControl = new TCFrequeControl();
        mLikeFrequeControl.init(2, 1);
    }
    if (mLikeFrequeControl.canTrigger()) {
        mHeartCount++;
        updatePraise();
        // 向后台发送点赞信息
        // mTCPlayerMgr.addHeartCount(mUserId);
        mTCPlayerMgr.sendPraiseToServer(mProgram.getStreamId());
    }
    break;
```

（2）执行测试

① 单击快捷工具栏中的“运行”按钮▶，运行程序。使用用户名“douxue1”、密码“123456789”登录，开始观看直播，并单击心形按钮♥，如图 6-2 所示。

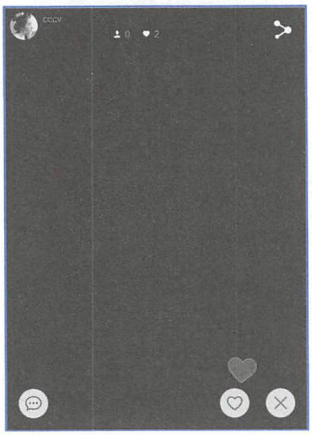


图 6-2
执行启动后的效果图

② 手动测试。

测试的场景（TestCase）如表 6-1 所示，可以编写单元测试或者自己手动测试。

表 6-1
测试用例表

测试场景	输入参数	预期结果
单击心形	—	出现飘心动画

经过测试，点赞功能能够正常使用。至此，本任务已经实现了点赞功能的开发。

项目实训

实训题目

练习 Android 响应事件的应用。

实训目的

掌握 Android 中响应事件的应用。

实训内容

新建一个项目，对一个按钮设置监听跳转至百度。

任务 6.2　实现弹幕模块

任务描述

在了解了弹幕工作的原理及应用后，调用腾讯提供的服务方法完成弹幕功能。

知识学习

用户可以输入弹幕让主播看到，主播也可以发送弹幕与观众互动，此处涉及观众界

非卖品，仅供非商业用途或交流学习使用

面与主播界面。

云通信提供单聊、群聊、资料托管、关系链托管、账号托管全方位解决方案，并提供完善的 APP 接入、后台管理接口。其架构图和服务原理如图 6-3 和图 6-4 所示。

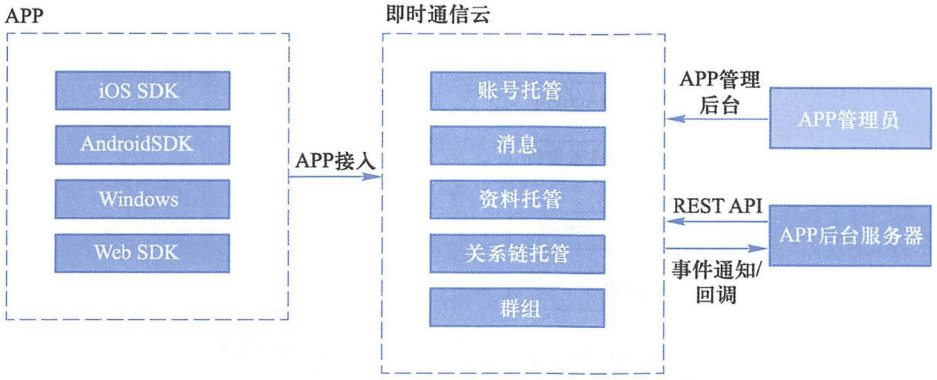


图 6-3
架构图

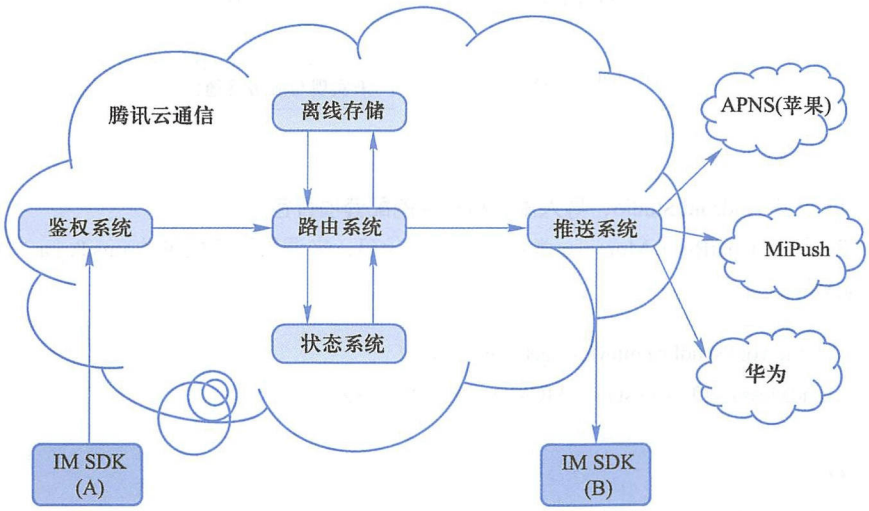


图 6-4
服务原理

腾讯云的通信系统类似于邮件系统，每条消息类似于一封邮件。

① A 发送的消息首先到鉴权系统，该系统负责对消息发送者的合法性进行确认。

② 之后消息被送达路由系统，该系统的正常运转依赖“状态系统”，因为它可以确认目标接收方是否在线（或者弱在线）。

③ 如果状态系统给出的结论是在线（或者弱在线），则路由系统直接将消息转给推送系统。

④ 如果对方是在线状态，消息会直接推送给对方，如果对方是弱在线，消息则会采用 APNS，华为或者小米的推送服务将消息推送给指定的推送服务系统。

⑤ 但如果“状态系统”给出的结论是目标接收方不在线，则消息直接落地到离线存储系统，等目标接收方上线后再次接收。

任务实现

弹幕功能具体流程如图 6-5 所示。

微课 6-2
实现弹幕模块

非卖品，仅供非商业用途或交流学习使用

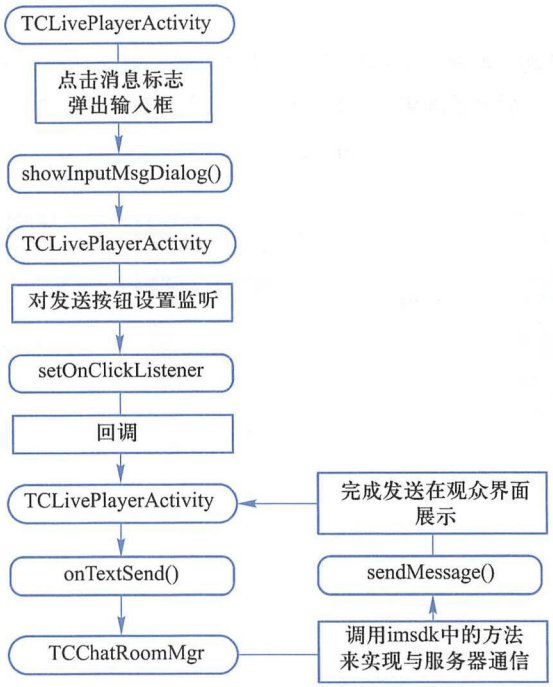


图 6-5
弹幕功能流程图

源码 6-2:
实现弹幕模块

（1）实现步骤

① 打开 Android Studio，导入本教材提供的配套项目包。

② 在 TCChatRoomMgr.java 类中加入如下代码，来调用腾讯提供的 imsdk.jar 文件中的 sendMessage()方法。

```
    public void sendDanmuMessage(String msg) {
    sendMessage(TCConstants.IMCMD_DANMU, msg);
}
/**
 * 发送消息
 *
 * @param cmd    控制符（代表不同的消息类型）具体查看 TCContants.IMCMD 开头变量
 * @param param  参数
 */
private void sendMessage(int cmd, String param) {

    sendMessage(cmd, param, new TIMValueCallBack<TIMMessage>() {
        @Override
        public void onError(int i, String s) {
            if (null != mTCChatRoomListener)
                mTCChatRoomListener.onSendMsgCallback(-1, null);
        }
```

非卖品，仅供非商业用途或交流学习使用


```
            @Override
            public void onSuccess(TIMMessage timMessage) {
                if (null != mTCChatRoomListener)
                    mTCChatRoomListener.onSendMsgCallback(0, timMessage);
            }
        });
    }
        private void sendMessage(int cmd, String param, TIMValueCallBack<TIMMessage>
timValueCallBack) {

            JSONObject sendJson = new JSONObject();
            try {
                sendJson.put("userAction", cmd);
                sendJson.put("userId", TCUserInfoMgr.getInstance().getUserId());
                sendJson.put("nickName", TCUserInfoMgr.getInstance().getNickname());
                sendJson.put("headPic", TCUserInfoMgr.getInstance().getHeadPic());
                sendJson.put("msg", param);
            } catch (JSONException e) {
                e.printStackTrace();
            }
            String cmds = sendJson.toString();
            Log.i(TAG, "send cmd : " + cmd + "|" + cmds);

            TIMMessage msg = new TIMMessage();

    //          if (cmd == TCConstants.IMCMD_PAILN_TEXT) {

                TIMTextElem elem = new TIMTextElem();
                elem.setText(cmds);

                if (msg.addElement(elem) != 0) {
                    Log.d(TAG, "addElement failed");
                    return;
                }

            sendTIMMessage(msg, timValueCallBack);

        }
    /**
     * 发送消息
```

非卖品，仅供非商业用途或交流学习使用


```
     *
     * @param msg                    TIM 消息
     * @param timValueCallBack  发送消息回调类
     */
    public void sendTIMMessage(TIMMessage msg, TIMValueCallBack<TIMMessage>
timValueCallBack) {
        if (mGroupConversation != null)
            mGroupConversation.sendMessage(msg, timValueCallBack);
    }
```

③ 观众界面，在 TCLivePlayerActivity.java 类中，设置点击发言标志的图标时展示输入框，代码如下。

```
    case R.id.btn_message_input:
        showInputMsgDialog();
        break;

    private void showInputMsgDialog() {
        WindowManager windowManager = getWindowManager();
        Display display = windowManager.getDefaultDisplay();
        WindowManager.LayoutParams lp = mInputTextMsgDialog.getWindow().getAttributes();

        lp.width = (display.getWidth()); //设置宽度
        mInputTextMsgDialog.getWindow().setAttributes(lp);
        mInputTextMsgDialog.setCancelable(true);

        mInputTextMsgDialog.getWindow().setSoftInputMode(WindowManager.LayoutParams.
SOFT_INPUT_STATE_VISIBLE);
        mInputTextMsgDialog.show();
    }
```

④ 在 joinRoom 方法中加入如下代码来初始化加载弹幕控件。

```
mTCChatRoomMgr = TCChatRoomMgr.getInstance();
```

⑤ 对发言按钮设置监听，代码如下。

```
mInputTextMsgDialog.setmOnTextSendListener(this);
```

⑥ 在 TCInputTextMsgDialog.java 类中，对发送按钮设置监听，代码如下。

```
confirmBtn.setOnClickListener(new View.OnClickListener() {
    @Override
```

非卖品，仅供非商业用途或交流学习使用


```
        public void onClick(View view) {
            String msg = messageTextView.getText().toString().trim();
            if (!TextUtils.isEmpty(msg)) {

                mOnTextSendListener.onTextSend(msg, mDanmuOpen);
                imm.showSoftInput(messageTextView, InputMethodManager.SHOW_FORCED);

                imm.hideSoftInputFromWindow(messageTextView.getWindowToken(), 0);
                messageTextView.setText("");
                dismiss();
            } else {
                Toast.makeText(mContext, "input can not be empty!", Toast.LENGTH_LONG).
show();
            }
            messageTextView.setText(null);
        }
    });
```

⑦ 在 TCLivePlayerActivity.java 类中，加入如下方法来处理消息发送成功后的结果，代码如下。

```
    public void onSendMsgCallback(int errCode, TIMMessage timMessage) {
        //消息发送成功后回显
        if (errCode == 0) {
            TIMElemType elemType = timMessage.getElement(0).getType();
            if (elemType == TIMElemType.Text) {
                Log.d(TAG, "onSendTextMsgsuccess:" + errCode);
            } else if (elemType == TIMElemType.Custom) {
                //custom 消息存在消息回调，此处显示成功/失败
                Log.d(TAG, "onSendCustomMsgsuccess:" + errCode);
            }
        } else {
            Log.d(TAG, "onSendMsgfail:" + errCode);
        }
    }
```

⑧ 请模仿观众模块的弹幕，完成主播模块的弹幕。

（2）执行测试

① 单击快捷工具栏中的“运行”按钮▶，运行程序。使用用户名“douxue1”、密码“123456789”登录，开始观看直播发送弹幕，如图 6-6 所示。

非卖品，仅供非商业用途或交流学习使用

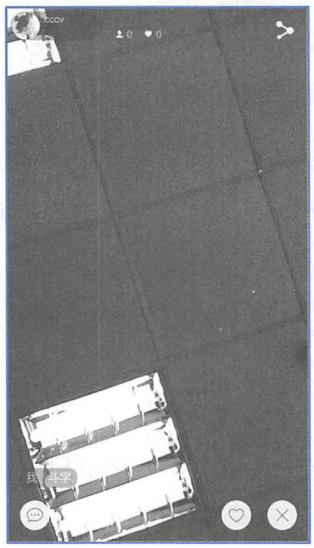


图 6-6
执行启动后的效果图

② 手动测试。

测试的场景（TestCase）如表 6-2 所示，可以编写单元测试或者自己手动测试。

表 6-2
测试用例表

测试场景	输入参数	预期结果
发送弹幕	斗学	左下角出现弹幕

经过测试，弹幕功能能够正常使用。至此，本任务已经实现了弹幕功能的开发。

项目实训

实训题目

调研腾讯云中的云通信服务。

实训目的

了解腾讯云通信服务的工作原理及使用方式。

实训内容

在腾讯云官网上查找资料并编写文档。

任务 6.3　实现功能扩展

任务描述

本任务在读者了解摄像头的调用方法后，完成摄像头的反转功能与美颜美白功能，使本系统进一步完善。

非卖品，仅供非商业用途或交流学习使用

知识学习

下面介绍摄像头的调用。

（1）应用场景

在开发 Android 系统 App 的过程中，有时需要调用手机自身设备的功能，本文侧重摄像头拍照功能的调用。

（2）知识点介绍

使用权限：调用手机自身设备功能（摄像头拍照功能），应该确保已经在 AndroidManifest.xml 中正确声明了对摄像头的使用及其他相关的 feature。

```
<!--摄像头权限-->
<uses-permission android:name="android.permission.CAMERA"/>
<!--存储权限 SD 卡读写权限-->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<!--Camera Feature 声明对 camera feature 的使用-->
<uses-feature android:name="android.hardware.camera"/>
<uses-feature
android:name="android.hardware.camera.autofocus"
android:required="false"/>
<!--视频、录音权限请求音频捕获权限-->
<uses-permission android:name="android.permission.RECORD_VIDEO"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

（3）Class 类

① SurfaceView。

视图（View）的继承类，这个视图里内嵌了一个专门用于绘制的 Surface。开发者可以控制这个 Surface 的格式和尺寸。Surfaceview 控制这个 Surface 的绘制位置。

② SurfaceHolder。

SurfaceHolder 是控制 Surface 的一个抽象接口，开发者可以通过 SurfaceHolder 来控制 Surface 的尺寸和格式，或者修改 Surface 的像素，监视 Surface 的变化等，SurfaceHolder 是 SurfaceView 的典型接口。

③ SurfaceHolder.Callback。

用户可以实现此接口接收 Surface 变化的消息。当用在一个 SurfaceView 中时，它只在 SurfaceHolder.Callback.surfaceCreated()和 SurfaceHolder.Callback.surfaceDestroyed()之间有效。设置 Callback 的方法是 SurfaceHolder.addCallback。

任务实现

功能拓展功能具体流程如图 6-7 所示。

微课 6-3
实现功能扩展模块

非卖品，仅供非商业用途或交流学习使用

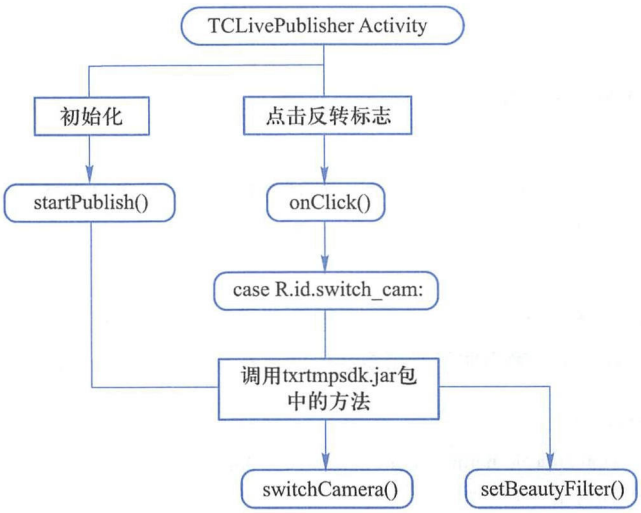


图 6-7
功能拓展流程图

源码 6-3:
实现功能扩展

打开 Android Studio，导入本教材提供的配套项目包。

1. 摄像头反转

设置摄像头直播的摄像头反转，代码如下。

```
case R.id.switch_cam:
    mTXLivePusher.switchCamera();
    break;
```

执行效果如图 6-8 所示。

图 6-8
执行启动后的效果图

2. 美颜、美白

美颜、美白的使用，在直播初始化的时候加入如下代码。

```
if(!mTXLivePusher.setBeautyFilter(TCUtils.filtNumber(9, 100, mBeautyLevel),
    TCUtils.filtNumber(3, 100, mWhiteningLevel))) {
     Toast.makeText(getApplicationContext(), "当前机型的性能无法支持美颜功能",
    Toast.LENGTH_SHORT).show();
}
```

执行效果如图 6-9 所示。

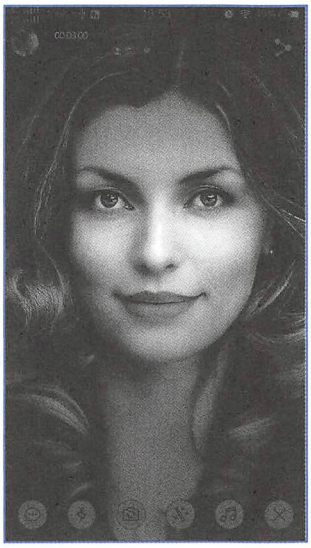


图 6-9
执行启动后的效果图

项目实训

实训题目

练习闪光灯接口的使用。

实训目的

了解如何调用手机的闪光灯接口。

实训内容

在网上查找如何调用闪光灯的资料，并实现闪光灯接口的使用。

单元小结

本单元完成了直播互动模块的点赞功能与弹幕功能，以及直播界面的一些优化功能。通过对这些功能的开发，相信读者在学习了本单元之后对 Android 开发会有进一步的理解。

单元 7

直播后台管理

学习目标

PPT7:
直播后台管理

知识目标

- 进一步了解 MD5。
- 了解直播后台管理的原理。
- 了解 MyBatis 框架。

技能目标

- 掌握开启直播功能的实现方法。
- 掌握直播列表功能的实现方法。
- 掌握观看直播功能的实现方法。

学习情境

斗学直播 APP 在小赵的努力下已经完成了 APP 客户端的视频直播功能，考虑到一些视频直播内容可能存在违规违纪的情况，公司想在后台添加一个视频直播内容监控的平台，能够实现监控当前正在直播和支持视频回放直播间的视频内容，能够对直播间进行封禁和对用户进行封禁，被封禁的直播间将不能再进行视频直播，被封禁的用户将不能登录 APP 客户端。公司将需求告诉了工程师小赵，于是小赵决定在后台添加直播管理和用户管理来实现公司需求。

任务 7.1　实现登录注销功能

任务描述

实现一个校验用户名、密码是否正确的方法。登录信息正确，登录后台系统；登录信息错误，提示相关错误信息。

知识学习

用户存储在数据库中的账户信息都是需要被加密过的，比如用户登录密码。为了保护数据的安全性，一般不会将密码以明文的形式存放在数据库中。本系统在用户注册和登录验证时采用 MD5 进行加密。

MD5 即 Message-Digest Algorithm 5（信息—摘要算法 5），用于确保信息传输完整一致，是计算机广泛使用的杂凑算法之一（又译摘要算法、哈希算法），主流编程语言普遍已有 MD5 实现。将数据（如汉字）运算为另一固定长度值，是杂凑算法的基础原理，MD5 的前身有 MD2、MD3 和 MD4。

MD5 算法具有以下特点。

① 压缩性：任意长度的数据，算出的 MD5 值长度都是固定的。

② 容易计算：从原数据计算出 MD5 值很容易。

③ 抗修改性：对原数据进行任何改动，哪怕只修改一个字节，所得到的 MD5 值都有很大区别。

④ 强抗碰撞：已知原数据和其 MD5 值，想找到一个具有相同 MD5 值的数据（即伪造数据）是非常困难的。

MD5 的作用是让大容量信息在用数字签名软件签署私人密钥前被“压缩”成一种保密的格式（就是把一个任意长度的字节串变换成一定长度的十六进制数字串）。除了 MD5 以外，其中比较有名的还有 sha-1、RIPEMD 及 Haval 等。

MD5 加密是日常开发中较为常用的一种加密手段，开发人员可以根据项目的安全需求灵活去变动，增加破译难度。

微课 7-1
实现登录注销功能

任务实现

登录注销功能具体流程如图 7-1 所示。

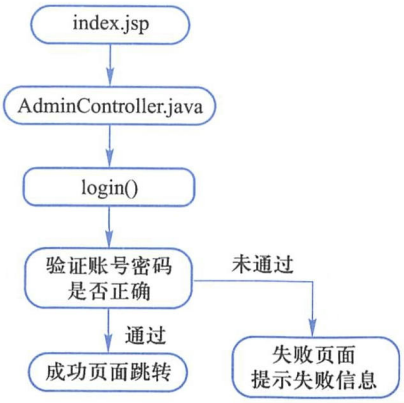


图 7-1
登录注销流程图

① 打开 Eclipse 导入本教材提供的配套项目包。

② 用户登录控制层。

源码 7-1：
实现登录注销功能

在 src/main/java/com/douxue/live/controller/web/admin/AdminController.java 文件中添加处理前端界面请求的方法，具体实现代码如下。

```
    /**
     * 登录
     * @param account
     * @param password
     * @return
     */
    @RequestMapping(value = "login", method = RequestMethod.POST)
    @ResponseBody
    public RestResult login(@RequestParam("account") String account, @RequestParam
("password") String password) {
        AdminUser admin = new AdminUser();
        admin.setAccount(account);
        admin.setPassword(MD5Utils.encryptUserPassword(password));
        admin.setState(0);
        AdminUser loginAdmin = adminUserService.findUser(admin);
        if (null != loginAdmin) {
            SessionUtils.saveAdminUser(loginAdmin);
            return RestResult.OK;
        }
        return RestResult.FAIL;
    }
```

③ 用户登录服务层。

在 src/main/java/com/douxue/live/service/user/impl/UserServiceImpl.java 文件中添加用户登录校验方法，具体实现代码如下。

```
    /**
```

```
     * 用户登录
     */
    @Override
    public RestResult login(String account, String password) {
        User query = new User();
        query.setAccount(account);
        query.setPassword(MD5Utils.encryptUserPassword(password));
        User user = userMapper.findOneUser(query);
        if (null == user) {
            return new RestResult(ErrorCode.ERROR_ACCOUNT_OR_PWD);
        }
        if(user.getState().intValue()== EnumUser.STATE_STOP.getValue()){
            return new RestResult(ErrorCode.ERROR_ACCOUNT_STOP);
        }
        user.setPassword(null);
        user.setWxUnionid(null);
        return new RestResult(user);
    }
```

④ 用户登录数据层。

在 src/main/java/com/douxue/live/dao/mapper/UserMapper.java 文件中添加登录验证接口，具体实现代码如下。

```
User findOneUser(User user);
```

在 src/main/resources/mapper/UserMapper.xml 文件中添加登录验证动态 SQL，具体实现代码如下。

```
    <select id="findOneUser" resultMap="BaseResultMap" parameterType="com.douxue.live.dao.
entity.User">
        SELECT
        <include refid="Base_Column_List" />
        FROM t_user
        <where>
            <if test="wxUnionid != null">
                AND wx_unionid = #{wxUnionid}
            </if>
            <if test="accountType != null">
                AND account_type = #{accountType}
            </if>
            <if test="userId != null">
                AND user_id = #{userId}
            </if>
```


版权所有，严禁传播，违者自负法律责任！

```
            <if test="password != null">
                AND password = #{password}
            </if>
            <if test="account != null">
                AND account = #{account}
            </if>
            <if test="nickName != null">
                AND nick_name = #{nickName}
            </if>
            <if test="userName != null">
                AND user_name = #{userName}
            </if>
            <if test="state != null">
                AND state = #{state}
            </if>
        </where>
        LIMIT 1
    </select>
```

⑤ 用户登录视图层。

在 webapp/WEB-INF/views/pages/文件夹下新建文件 index.jsp。具体登录界面代码如下。

```
    <div class="login_right_div">
        <form class="bs-example bs-example-form data" role="form">
            <div class="input-group form-group">
                <span class="input-group-addon">账   号</span>
                <input type="text" id="account" name="account" class="form-control" placeholder=
"管理员账号">
            </div>
            <div class="input-group form-group">
                <span class="input-group-addon">密   码</span>
                <input type="password" id="password" name="password" class="form-control"
placeholder="管理员密码">
            </div>
            <div class="text-center">
                <button type="submit" class="btn btn-primary btn-sm">登   
录</button>
            </div>
        </form>
    </div>
```

执行效果如图 7-2 所示。

版权所有，严禁传播，违者自负法律责任！

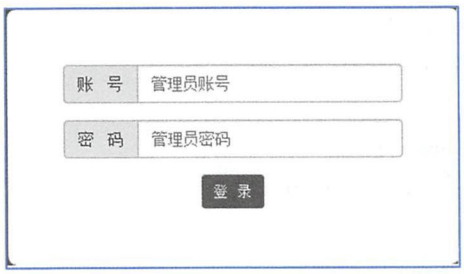


图 7-2
登录界面

⑥ 数据交互。

在上一小节内容中，已经完成了登录界面的编写，接下来就要与后端进行数据交互，本任务中采用 Ajax 与 JSON 来实现数据交互。

在界面代码 index.jsp 中，找到 id="login_right_div"的 div 模块为其添加单击监听事件方法，如下所示。

```
function login(callback){
    var account = $("#account").val();
    var password = $("#password").val();
    if(!account || !password){
        layer.msg("请输入账号与密码");
    }else{
        var data = {
            "account":account,
            "password":password
        }
        serverLogin(data,function(rs){
            callback(rs);
        });
    }
}
function serverLogin(data,callback){
    data_ajax(
        {
            "url":"<%=contextPath%>/web/admin/login",
            "data":data
        },
        function(rs){
            callback(rs);
        }
    );
}
$(".login_right_div").on("submit","form",function(){
    login(function(rs){
        //登录成功
```


版权所有，严禁传播，违者自负法律责任！

```
            window.location.href = "<%=contextPath%>/web/admin/home";
        });
        return false;
    });
```

⑦ 执行效果。

将项目部署到 Tomcat 中并发布，访问 http://localhost:8080/douxue_web，验证登录，如图 7-3 所示。

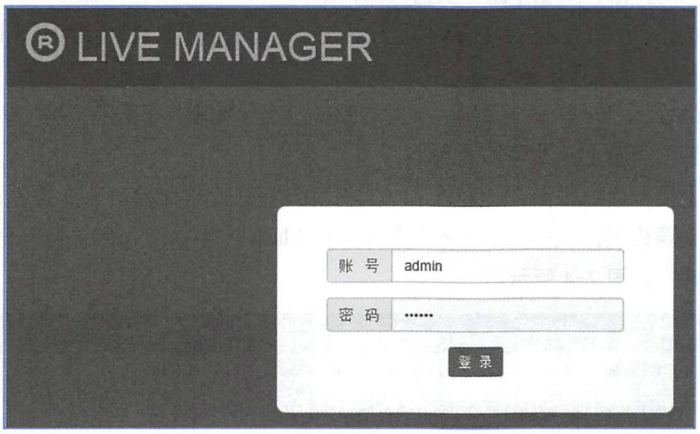


图 7-3
登录系统

项目实训

实训题目

在后台登录和注销账户。

实训目的

1. 了解 MD5 加密的原理。
2. 掌握前后台的基本通信手段。

实训内容

实现用户后台注销功能。

任务 7.2　实现直播管理功能

任务描述

实现对直播间进行监控和控制。从数据库中取出直播信息，并能够显示直播状态、支持查看视频直播内容、支持设置直播状态，达到禁播的目的；能够根据不同的字段筛选出对应的直播信息列表。

知识学习

腾讯云 Web 播放器主要解决在手机浏览器和 PC 浏览器上播放音/视频流的问题，使

版权所有，严禁传播，违者自负法律责任！

视频内容可以不依赖用户安装 APP 就可以在朋友圈、微博等社交平台进行传播。

直播是指视频源是实时的，一旦主播停播了，这个地址就失去了意义，而且由于是实时直播，所以播放器在播直播视频的时候是没有进度条的。

点播是指视频源是一个服务器上的文件，文件只要没有被提供方删除，就可以随时播放，而且由于整个视频都在服务器上，所以播放器在播点播视频的时候是有进度条的。

Web 播放器的视频播放能力本身不是通过网页代码实现的，而是靠浏览器的支持，所以其兼容性并不好，必须要接受一个事实：不是所有的手机浏览器都能有符合预期的表现，有些手机浏览器甚至根本就不支持视频播放。

最常见的用于网页直播的视频源地址是以 m3u8 结尾的地址，被称为 HLS（HTTP Live Streaming），这是苹果推出的标准。由于苹果的影响力，目前各手机浏览器产品对这种格式的兼容性最好，但它有一个天然的问题，就是延迟比较大，一般有 20～30 秒左右的延迟，但在手机浏览器上并没有其他选择。

在 PC 上情况会好很多，因为 PC 上的浏览器目前还没有抛弃 Flash 控件，而 Flash 控件支持的视频源格式比较多，而且各浏览器上的 Flash 控件都是 Adobe 自己开发的，所以兼容性非常好，如图 7-4 所示。

视频协议	用途	示例地址	PC浏览器	移动浏览器
HLS(m3u8)	可用于直播	http://2157.liveplay.myqcloud.com/2157_358535a.m3u8	支持	支持
HLS(m3u8)	可用于点播	http://200002949.vod.myqcloud.com/200002949_b6ffc.f0.m3u8	支持	支持
FLV	可用于直播	http://2157.liveplay.myqcloud.com/2157_358535a.flv	支持	不支持
FLV	可用于点播	http://200002949.vod.myqcloud.com/200002949_b6ffc.f0.flv	支持	不支持
RTMP	只适用直播	rtmp://2157.liveplay.myqcloud.com/live/2157_280d88	支持	不支持
MP4	只适用点播	http://200002949.vod.myqcloud.com/200002949_b6ffc.f0.MP4	支持	支持

图 7-4
视频协议

微课 7-2
实现直播管理功能

任务实现

直播管理功能具体流程如图 7-5 所示。

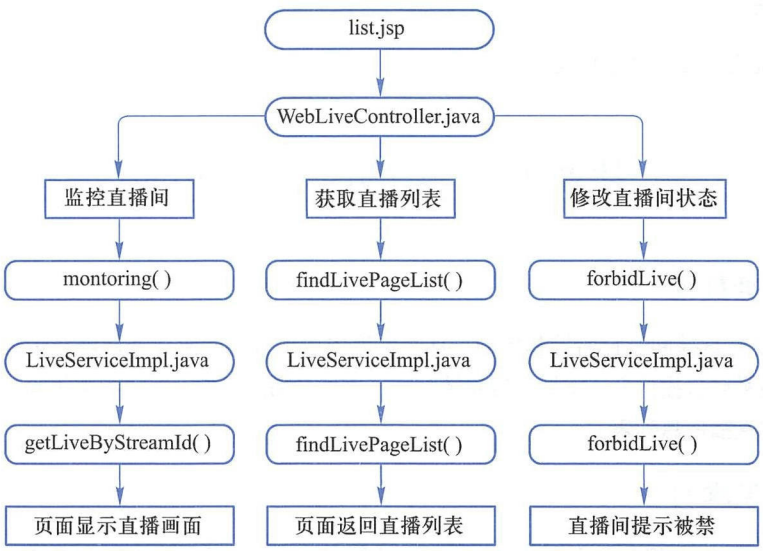


图 7-5
直播管理流程图

版权所有，严禁传播，违者自负法律责任！

1. 添加直播列表

源码 7-2：实现直播管理功能

① 打开 Eclipse，导入本教材提供的配套项目包。

② 获取直播列表控制层。

在 src/main/java/com/douxue/live/controller/web/live/WebLiveController.java 文件中添加获取直播列表界面请求的方法，具体实现代码如下。

```
/**
* 查询直播数据分页列表
*
* @param pageNum
* @param pageSize
* @param live
* @return
*/
@RequestMapping("findLivePageList")
@ResponseBody
public RestResult findLivePageList(@RequestParam(value = "pageNum", defaultValue =
"1")    Integer pageNum,
    @RequestParam(value = "pageSize", defaultValue = "10") Integer pageSize, Live live) {
    PageData<Live> pageData = liveService.findLivePageList(pageNum, pageSize, live);
    return new RestResult(pageData);
}
```

③ 获取直播列表服务层。

在 src/main/java/com/douxue/live/service/live/impl/LiveServiceImpl.java 文件中添加查询直播数据列表方法，代码如下。

```
/**
 * 查询直播数据分页列表
 */
@Override
public PageData<Live> findLivePageList(Integer pageNum, Integer pageSize, Live live) {
    PageParam<Live> param = new PageParam<Live>(live, pageNum, pageSize);
    List<Live> data = liveMapper.findLivePageList(param);
    if (null != data && data.size() > 0) {
        for (Live l : data) {
            l = setLiveRedisData(l);
        }
    }
    PageData<Live>page=new PageData<Live>(pageNum, pageSize, param.getDataTotal(),data);
    return page;
}
```


版权所有，严禁传播，违者自负法律责任！

④ 获取直播列表数据层。

在 src/main/java/com/douxue/live/dao/mapper/LiveMapper.java 文件中添加查询直播接口，具体实现代码如下。

```
List<Live> findLivePageList(PageParam<Live> param);
```

在文件中添加查询直播动态 SQL，具体实现代码如下。

```
<select id="findLivePageList" resultMap="BaseResultMap" parameterType="com.douxue.live.common.page.PageParam">
    SELECT
    l.*
    FROM t_live l INNER JOIN t_user u ON l.user_id = u.user_id
    WHERE l.state != -1
      <if test="p.userId != null and p.userId != ''">
        AND l.userId = #{p.userId}
      </if>
      <if test="p.streamId != null and p.streamId != ''">
        AND l.stream_id = #{streamId}
      </if>
      <if test="p.title != null and p.title != ''">
        AND l.title LIKE CONCAT('%',#{p.title},'%')
      </if>
      <if test="p.state != null">
        AND l.state = #{p.state}
      </if>
      <if test="p.user.nickName != null and p.user.nickName != ''">
        AND u.nick_name LIKE CONCAT('%',#{p.user.nickName},'%')
      </if>
    ORDER BY l.create_time DESC
</select>
```

⑤ 直播列表视图层。

在 webapp/WEB-INF/views/pages/live 文件夹下新建文件 list.jsp，界面代码如下。

```
<div style="width: 100%;margin-left: 2px;">
    <form id="search_form">
        <input type="hidden" id="pageNum" name="pageNum" />
        <input type="hidden" name="pageSize" value="10" />
        <div class="row">
            <div class="col-lg-2">
                <label>直播间:</label>
                <input type="text" name="title" class="form-control" placeholder="直播间名称(模糊查询)" />
```


版权所有，严禁传播，违者自负法律责任！

```
                </div>
                <div class="col-lg-2">
                    <label>用户昵称:</label>
                    <input type="text" name="user.nickName" class="form-control" placeholder=
"用户昵称" />
                </div>
                <div class="col-lg-2">
                    <label>状态:</label>
                    <select class="form-control" name="state">
                        <option value="">全部</option>
                        <option value="0">已结束</option>
                        <option value="1">直播中</option>
                        <option value="100">回放</option>
                        <!-- <option value="-1">初始化</option> -->
                    </select>
                </div>
                <div class="col-lg-2">
                    <br />
                    <button type="submit" id="search_btn" class="btn btn-primary btn-lg" style=
"font-size: 9px">
                        <span class="glyphicon glyphicon-search"></span>搜索
                    </button>
                </div>
            </div>
        </form>
    </div>
    <div style="width: 100%;margin-left: 2px;margin-top: 20px;height:430px;">
        <table class="table table-hover" id="data_table">
            <thead>
                <tr style="background-color: #E0E0E0">
                    <th>直播间 ID</th>
                    <th>直播间</th>
                    <th>用户</th>
                    <th>点赞数</th>
                    <th>观看数</th>
                    <th>状态</th>
                    <th>创建时间</th>
                    <th>操作</th>
                </tr>
            </thead>
            <tbody>
            </tbody>
```


版权所有，严禁传播，违者自负法律责任！

```
        </table>
    </div>
    <div id="laypage"></div>
```

执行效果如图 7-6 所示。

图 7-6
直播管理

⑥ 数据交互。

在上一小节中，已经完成了直播列表界面代码的编写，接下来就要与后端进行数据交互，本任务采用 Ajax 与 JSON 来实现数据交互。

在界面代码 list.jsp 中，添加直播列表初始化方法来获取直播列表，交互代码如下所示。

```
//页面初始化
list(1,function(data){
    //分页
    groPage('laypage',data.data.pageTotal,5,function(curr){
        list(curr,function(data){});
    });
});
//表格数据初始化
function list(pageNum,callback){
    $('#pageNum').val(pageNum);
    var data = $('#search_form').serialize();
    data_ajax(
        {
            "url":'<%=contextPath%>/web/live/findLivePageList',
            "data":data
        },
        function(rs){
            var _tbody = $('#data_table tbody');
            _tbody.empty();
            var html = '';
```


版权所有，严禁传播，违者自负法律责任！

```
                for(var key in rs.data.pageData){
                    var data = rs.data.pageData[key];
                    html += "<tr>";
                    html += "<td>" + NULL(data.streamId) + "</td>";
                    html += "<td>" + NULL(data.title) + "</td>";
                    html += "<td>" + NULL(data.user.nickName) + "</td>";
                    html += "<td>" + NULL(data.praiseCount) + "</td>";
                    html += "<td>" + NULL(data.viewerCount) + "</td>";
                    html += "<td>" + NULL(data.stateName) + "</td>";
                    html += "<td>" + NULL(data.createTime) + "</td>";
                    html += "<td>";
                    html += '<a href="javascript:void(0)" class="btn btn-link btn-xs monitoring"
data_id="'+data.streamId+'">监控</a>';
                    html += '<a href="javascript:void(0)" class="btn btn-link btn-xs stop"
data_id="'+data.streamId+'">禁播</a>';
                    html += "</td>";

                    html += "</tr>";
                }
                _tbody.append(html);
                callback(rs);
            }
        );
    }
```

⑦ 执行效果。

将项目部署到 Tomcat 中并发布，访问 http://localhost:8080/douxue_web/web/admin/home，登录并在直播管理中查看直播列表，如图 7-7 所示。

图 7-7
直播列表

2. 添加视频监控功能

① 视频内容监控控制层。

在 src/main/java/com/douxue/live/controller/web/live/WebLiveController.java 文件中添加根据直播的 streamId 获取直播内容请求的方法，具体实现代码如下。

```
/**
 * 根据 streamId 获取直播
 * @param request
 * @param response
 * @return
 */
@RequestMapping("monitoring")
public ModelAndView monitoring(HttpServletRequest request, HttpServletResponse response) {
    ModelAndView view = new ModelAndView("/pages/live/monitoring");
    String streamId = request.getParameter("streamId");
    Live live = null;
    if (null != streamId) {
        live = liveService.getLiveByStreamId(streamId);
    }
    view.addObject("live", live);
    return view;
}
```

② 视频内容监控服务层。

在 src/main/java/com/douxue/live/service/live/impl/LiveServiceImpl.java 文件中添加查询直播方法，代码如下。

```
/**
 * 根据 streamId 获取直播
 */
@Override
public Live getLiveByStreamId(String streamId) {
    Live live = liveMapper.selectByPrimaryKey(streamId);
    live = setLiveRedisData(live);
    return live;
}
```

③ 视频内容监控数据层。

在 src/main/java/com/douxue/live/dao/mapper/LiveMapper.java 文件中添加查询直播接口，代码如下。

```
Live selectByPrimaryKey(String streamId);
```

在 src/main/resources/mapper/LiveMapper.xml 文件中添加查询直播动态 SQL，代码如下。

```
<select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.String" >
select
<include refid="Base_Column_List" />
from t_live
where stream_id = #{streamId,jdbcType=VARCHAR}
</select>
```

④ 视频内容监控视图层。

在 webapp/WEB-INF/views/pages/live 文件夹下新建文件 monitoring.jsp，界面代码如下。

```
<div class="page-content">
    <!--播放器层-->
    <div id="id_video_container"></div>
</div>
```

⑤ 数据交互。

在上一小节中，已经完成了直播监控界面代码的编写，接下来就要与后端进行数据交互，本任务采用 Ajax 与 JSON 来实现数据交互。

在界面代码 list.jsp 中，交互代码如下所示。

```
// 监控
$('#data_table').on('click','.monitoring',function(){
    var _self = $(this);
    var id = _self.attr('data_id');
    window.open("<%= contextPath %>/web/live/monitoring?streamId=" + id);
});
```

在获取到直播内容后将会跳转至 monitoring.jsp 页面，获取直播内容，代码如下：

```
window.addEventListener("load",function(){
    var streamId = "${live.streamId}";
    //视频状态
    var state = 1;
    server(
        {
            "url":"<%=contextPath%>/web/live/getLiveByStreamId",
            "data":{
                "streamId":streamId
            }
        },
        function(data){
            if(data.code != 1){
                layer.open({
```

```
                content:data.msg,
                time:2
            });
        }else{
            if(!data.data){
                layer.open({
                    content:"服务器错误"
                });
                return;
            }
            state = data.data.state;
            var options = {};
            //直播与回放 js 库
            var $script = document.createElement("script");
            var $scriptSrc = "";
            if(state == 1){
                //直播
                $scriptSrc="//qzonestyle.gtimg.cn/open/qcloud/video/live/h5/live_connect.js";
                options = {
                    "live_url":data.data.playHlsUrl,
                };
            }else{
                //回放
                $scriptSrc = "//qzonestyle.gtimg.cn/open/qcloud/video/h5/h5connect.js";
                if(!data.data.liveVideos){
                    layer.open({
                        content:"当前直播已结束，请等待回放"
                    });
                    return;
                }
                options = {
                    "third_video": {
                        "urls":{
                            20 : data.data.liveVideos[0].videoUrl
                        }
                    }
                };
            }
            //播放器初始化
            var $script = document.createElement("script");
            $script.src = $scriptSrc;
```

```
                document.body.appendChild($script);
                $script.onload = function(){
                    new player_content().init(options);
                }
            }
        }
    );
});
```

⑥ 执行效果。

将项目部署到 Tomcat 中并发布，访问 http://localhost:8080/douxue_web/web/admin/home，在直播列表中选择状态为直播中的信息，单击“监控”按钮查看直播内容，如图 7-8 所示。

图 7-8 监控

3. 添加禁播功能

① 禁播视频直播控制层。

在 src/main/java/com/douxue/live/controller/web/live/WebLiveController.java 文件中添加根据直播的 streamId 禁播的请求方法，具体实现代码如下。

```
/**
 * 禁播直播
 *
 * @param streamId
 * @return
```

```
     */
    @RequestMapping(value = "forbidLive", method = RequestMethod.POST)
    @ResponseBody
    public RestResult forbidLive(@RequestParam("streamId") String streamId) {
        return liveService.forbidLive(streamId);
    }
```

② 禁播视频直播服务层。

在 src/main/java/com/douxue/live/service/live/impl/LiveServiceImpl.java 文件中添加禁播视频的方法，代码如下。

```
    /**
     * 禁播直播
     */
    @Override
    public RestResult forbidLive(String streamId) {
        // 根据直播 streamId 获取直播数据
        Live live = liveMapper.selectByPrimaryKey(streamId);
        if (null == live) {
            return new RestResult(ErrorCode.ERROR_NULL_DATA);
        }
        LinkedHashMap<String, Object> paramMap = new LinkedHashMap<String, Object>();
        // 设置签名过期时间
        long t = System.currentTimeMillis() / 1000 + TIMEOUT;
        // 拼接 API 参数
        paramMap.put("cmd", LIVE_APP_ID);
        paramMap.put("interface", "Live_Channel_SetStatus");
        paramMap.put("t", t);
        paramMap.put("sign", liveApiTool.getSign(t)); // 签名
        paramMap.put("Param.s.channel_id", streamId);
        paramMap.put("Param.n.status", 0); // 设置流状态 int 0:关闭； 1:开启
        String res = "";
        try {
            // 调用 API 强制断流
            res = HttpClientUtils.GET(LIVE_COMMON_ACCESS_URL, paramMap, 3000);
            logger.info("forbidLive - " + streamId + " - res : " + res);
            Map<String, Object> resMap = new HashMap<String, Object>();
            resMap = gson.fromJson(res, Map.class);
            // 断流结果:成功
            if (null != resMap.get("ret") && "0.0".equals(resMap.get("ret").toString())) {
                Live update = new Live();
                update.setStreamId(streamId);
```

```
                update.setState(EnumLive.STATE_FORBID.getValue());// 设置直播为禁播状态
                liveMapper.updateByPrimaryKeySelective(update); // 修改&保存直播状态
                return RestResult.OK;
            } else { // 断流结果:失败
                return new RestResult(0, resMap.get("message").toString());
            }
        } catch (IOException e) {
            logger.error("forbidLive - " + streamId + " FAIL");
            return RestResult.FAIL;
        }
    }
```

③ 禁播视频直播数据层。

在 src/main/java/com/douxue/live/dao/mapper/LiveMapper.java 文件中添加更新直播状态接口的代码，代码如下。

```
int updateByPrimaryKeySelective(Live record);
```

在 src/main/resources/mapper/LiveMapper.xml 文件中添加更新直播状态动态 SQL，代码如下。

```
<update id="updateByPrimaryKeySelective" parameterType="com.douxue.live.dao.entity.Live" >
update t_live
<set >
  <if test="userId != null" >
    user_id = #{userId,jdbcType=VARCHAR},
  </if>
  <if test="groupId != null" >
    group_id = #{groupId,jdbcType=VARCHAR},
  </if>
  <if test="title != null" >
    title = #{title,jdbcType=VARCHAR},
  </if>
  <if test="coverPic != null" >
    cover_pic = #{coverPic,jdbcType=VARCHAR},
  </if>
  <if test="location != null" >
    location = #{location,jdbcType=VARCHAR},
  </if>
  <if test="pushUrl != null" >
    push_url = #{pushUrl,jdbcType=VARCHAR},
  </if>
  <if test="playRtmpUrl != null" >
```

```
        play_rtmp_url = #{playRtmpUrl,jdbcType=VARCHAR},
      </if>
      <if test="playFlvUrl != null" >
        play_flv_url = #{playFlvUrl,jdbcType=VARCHAR},
      </if>
      <if test="playHlsUrl != null" >
        play_hls_url = #{playHlsUrl,jdbcType=VARCHAR},
      </if>
      <if test="playUrl != null" >
        play_url = #{playUrl,jdbcType=VARCHAR},
      </if>
      <if test="state != null" >
        state = #{state,jdbcType=INTEGER},
      </if>
      <if test="duration != null" >
        duration = #{duration,jdbcType=BIGINT},
      </if>
      <if test="startTime != null" >
        start_time = #{startTime,jdbcType=TIMESTAMP},
      </if>
      <if test="createTime != null" >
        create_time = #{createTime,jdbcType=TIMESTAMP},
      </if>
    </set>
    where stream_id = #{streamId,jdbcType=VARCHAR}
    </update>
```

④ 数据交互。

在上一小节中，已经完成了直播监控界面代码的编写，接下来就要与后端进行数据交互，本任务采用 Ajax 与 JSON 来实现数据交互。

在界面代码 list.jsp 中，交互代码如下所示。

```
// 禁播
$('#data_table').on('click','.stop',function(){
    var _self = $(this);
    layer.confirm('确定强制禁播该直播?',
        {
          btn: ['确定','取消'] //按钮
        }, function(){
            var id = _self.attr('data_id');
            $.post("<%= contextPath %>/web/live/forbidLive",{"streamId":id},function(rs){
                if(rs.code == 1){
```

```
                    layer.msg("已禁播",{icon:6});
                    $('#search_form').submit();
                } else {
                    layer.msg("失败",{icon:5});
                }
            });
        });
    });
```

⑤ 执行效果。

将项目部署到 Tomcat 中并发布，访问 http://localhost:8080/douxue_web/web/admin/home，在直播列表中选择状态为直播中的信息，单击“禁播”按钮后，查看 APP 的直播情况，如图 7-9 所示。

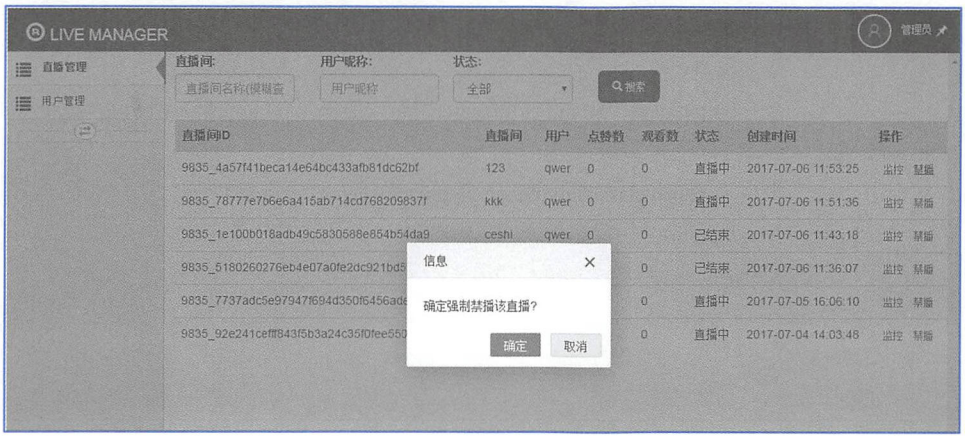


图 7-9 禁播

4. 添加搜索功能

① 搜索直播房间视图层。

在 webapp/WEB-INF/views/pages/live/list.jsp 中新增界面代码，代码如下。

```
<div style="width: 100%;margin-left: 2px;">
    <form id="search_form">
        <input type="hidden" id="pageNum" name="pageNum" />
        <input type="hidden" name="pageSize" value="10" />
        <div class="row">
            <div class="col-lg-2">
                <label>直播间:</label>
                <input type="text" name="title" class="form-control" placeholder="直播间名
称(模糊查询)" />
            </div>
            <div class="col-lg-2">
                <label>用户昵称:</label>
```

```
                    <input type="text" name="user.nickName" class="form-control" placeholder=
"用户昵称" />
                </div>
                <div class="col-lg-2">
                    <label>状态:</label>
                    <select class="form-control" name="state">
                        <option value="">全部</option>
                        <option value="0">已结束</option>
                        <option value="1">直播中</option>
                        <option value="100">回放</option>
                        <!-- <option value="-1">初始化</option> -->
                    </select>
                </div>
                <div class="col-lg-2">
                    <br />
                    <button type="submit" id="search_btn" class="btn btn-primary btn-lg" style=
"font-size: 9px">
                        <span class="glyphicon glyphicon-search"></span>搜索
                    </button>
                </div>
            </div>
        </form>
    </div>
```

② 数据交互。

在上一小节中，已经完成了搜索界面代码的编写，接下来就要与后端进行数据交互，本任务采用 Ajax 与 JSON 来实现数据交互。

在界面代码 list.jsp 中，交互代码如下所示。

```
//搜索
$('#search_form').on('submit',function(){
    list(1,function(data){
        //分页
        groPage('laypage',data.data.pageTotal,5,function(curr){
            list(curr,function(data){});
        });
    });
    return false;
});
```

③ 执行效果。

将项目部署到 Tomcat 中并发布，访问 http://localhost:8080/douxue_web/web/admin/

home，在“直播间”“用户昵称”和“状态”中设置搜索条件，单击“搜索”按钮搜索，如图 7-10 所示。

图 7-10 搜索

项目实训

实训题目

练习 Web 播放应用。

实训目的

1. 了解 Web 播放的基本原理。
2. 了解腾讯 Web 播放器的应用方法。

实训内容

实现对视频直播内容的监控。

任务 7.3　实现用户管理功能

任务描述

实现对用户进行统一的管理，可以查看用户的名称、状态等信息；实现可以将用户的账户禁用、限制账户登录，达到封禁和解封的能力。

知识学习

MyBatis 的强大特性之一便是它的动态 SQL。如果开发者有使用 JDBC 或其他类似框架的经验，就能体会到根据不同条件拼接 SQL 语句有多么痛苦。拼接的时候要确保不能忘了必要的空格，还要注意省掉列名列表最后的逗号。利用动态 SQL 这一特性可以彻底摆脱这种痛苦。

通常使用动态 SQL 不可能是独立的一部分，MyBatis 使用一种强大的动态 SQL 语言来改进这种情形，这种语言可以被用在任意的 SQL 映射语句中。

动态 SQL 元素和使用 JSTL 或其他类似基于 XML 的文本处理器相似。在 MyBatis 之前的版本中，有很多元素需要了解。MyBatis 3 大大提升了它们，现在用不到原先一半的元素就可以了。MyBatis 采用功能强大的基于 OGNL 的表达式来消除其他元素。

MyBatis 中用于实现动态 SQL 的元素主要有：if、choose(when，otherwise)、trim(where、set)、where、set、foreach。

任务实现

用户管理功能具体流程如图 7-11 所示。

微课 7-3
实现用户管理功能

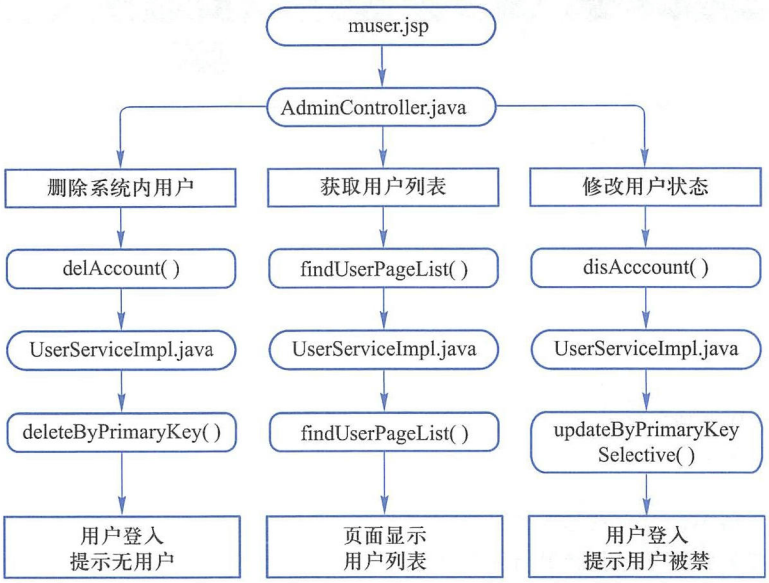


图 7-11
用户管理流程图

1. 添加用户管理列表

源码 7-3:
实现用户管理功能

① 打开 Eclipse，导入本教材提供的配套项目包。

② 分页获取用户列表控制层。

在 src/main/java/com/douxue/live/controller/web/admin/AdminController.java 文件中添加查找所有用户请求的方法，具体实现代码如下。

```
/**
 * 用户列表分页
 *
 * @param pageNum
 * @param pageSize
 * @param user
 * @return
 */
@RequestMapping("findUserPageList")
@ResponseBody
public RestResult findUserPageList(@RequestParam(value = "pageNum", defaultValue = "1")
Integer pageNum,
        @RequestParam(value = "pageSize", defaultValue = "10") Integer pageSize, User user) {
    PageData<User> pageData = userService.findUserPageList(pageNum, pageSize, user);
    return new RestResult(pageData);
}
```

③ 分页获取用户列表服务层。

在 src/main/java/com/douxue/live/service/user/impl/UserServiceImpl.java 文件中添加分页获取用户列表方法，具体实现代码如下。

```
/**
 * 用户列表
 */
@Override
public PageData<User> findUserPageList(Integer pageNum, Integer pageSize, User user) {
    PageParam<User> param = new PageParam<User>(user, pageNum, pageSize);
    List<User> data = userMapper.findUserPageList(param);
    PageData<User> page = new PageData<User>(pageNum, pageSize, param.getDataTotal(), data);
    return page;
}
```

④ 分页获取用户列表数据层。

在 src/main/java/com/douxue/live/dao/mapper/UserMapper.java 文件中添加分页获取用户列表接口的代码，具体实现代码如下。

```
List<User> findUserPageList(PageParam<User> param);
```

在 src/main/resources/mapper/UserMapper.xml 文件中添加根据条件查询用户动态 SQL，具体实现代码如下。

```
<select id="findUserPageList" resultMap="BaseResultMap" parameterType="com.douxue.live.
common.page.PageParam">
    SELECT
    u.*
    FROM t_user u
    WHERE u.account != 'admin'
    <if test="p.userId != null and p.userId != ''">
        AND u.userId = #{p.userId}
    </if>
    <if test="p.state != null">
        AND u.state = #{p.state}
    </if>
    <if test="p.nickName != null and p.nickName != ''">
        AND u.nick_name LIKE CONCAT('%',#{p.nickName},'%')
    </if>
    ORDER BY u.create_time DESC
</select>
```

⑤ 实现视图层界面。

在 webapp/WEB-INF/views/pages/live 文件夹下新建文件 muser.jsp。界面具体实现代

码如下。

```
<div style="width: 100%;margin-left: 2px;margin-top: 20px;height:430px;">
    <table class="table table-hover" id="data_table">
        <thead>
            <tr style="background-color: #E0E0E0">
                <th>用户 ID</th>
                <th>用户名</th>
                <th>昵称</th>
                <th>账户类型</th>
                <th>手机</th>
                <th>创建时间</th>
                <th>状态</th>
                <th>操作</th>
            </tr>
        </thead>
        <tbody>
        </tbody>
    </table>
</div>
<div id="laypage"></div>
```

⑥ 数据交互。

在上一小节中，已经完成了用户列表界面代码的编写，接下来就要与后端进行数据交互，本任务采用 Ajax 与 JSON 来实现数据交互。

在界面代码 muser.jsp 中，交互代码如下所示。

```
//页面初始化
list(1,function(data){
    //分页
    groPage('laypage',data.data.pageTotal,5,function(curr){
        list(curr,function(data){});
    });
});

//表格数据初始化
function list(pageNum,callback){
    $('#pageNum').val(pageNum);
    var data = $('#search_form').serialize();
    data_ajax(
        {
            "url":'<%=contextPath%>/web/admin/findUserPageList',
```


版权所有，严禁传播，违者自负法律责任！

```
                "data":data
            },
            function(rs){
                var _tbody = $('#data_table tbody');
                _tbody.empty();
                var html = '';
                for(var key in rs.data.pageData){
                    var data = rs.data.pageData[key];
                    html += "<tr>";
                    html += "<td>" + NULL(data.userId) + "</td>";
                    html += "<td>" + NULL(data.account) + "</td>";
                    html += "<td>" + NULL(data.nickName) + "</td>";
                    html += "<td>" + NULL(data.accountType) + "</td>";
                    html += "<td>" + NULL(data.phone) + "</td>";
                    html += "<td>" + NULL(data.createTime) + "</td>";
                    if(NULL(data.state) == -1){
                        html += "<td><i>已禁用</i></td>";
                        html += "<td>";
                        html += '<a href="javascript:void(0)" class="btn btn-link btn-xs open"
data_id="'+data.userId+'">启用</a>';
                    }else{
                        html += "<td>正常</td>";
                        html += "<td>";
                        html += '<a href="javascript:void(0)" class="btn btn-link btn-xs stop"
data_id="'+data.userId+'">禁用</a>';
                    }
                    html += '<a href="javascript:void(0)" class="btn btn-link btn-xs delete"
data_id="'+data.userId+'">删除</a>';
                    html += "</td>";

                    html += "</tr>";
                }
                _tbody.append(html);
                callback(rs);
            }
        );
    }
```

⑦ 执行效果。

将项目部署到 Tomcat 中并发布，访问 http://localhost:8080/douxue_web/web/admin/home，在用户管理中查看用户列表，如图 7-12 所示。

版权所有，严禁传播，违者自负法律责任！

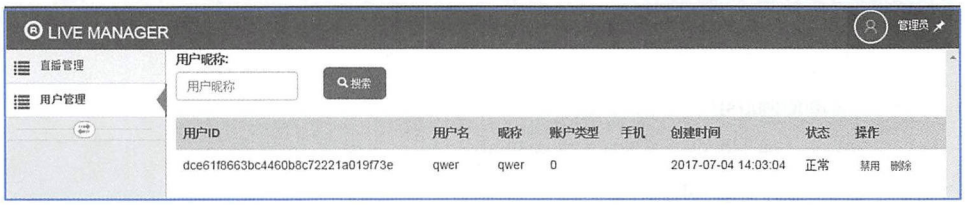


图 7-12
用户管理

2. 添加禁用和启用账号功能

① 禁用和启用账号控制层。

在 src/main/java/com/douxue/live/controller/web/admin/AdminController.java 文件中添加禁启用账户请求的方法，具体实现代码如下。

```
/**
 * 启/禁用账号
 *
 * @param userId
 * @return
 */
@RequestMapping(value = "disaccount", method = RequestMethod.POST)
@ResponseBody
public RestResult disAcccount(@RequestParam("userId") String userId, @RequestParam
("state") int state) {
    User record = new User();
    record.setUserId(userId);
    if (record != null) {
        record.setState(state); // 禁用状态
        int key = userService.updateByPrimaryKeySelective(record);
        if (key > 0) {
            return RestResult.OK;
        }
    }
    return RestResult.FAIL;
}
```

② 禁用和启用账号服务层。

在 src/main/java/com/douxue/live/service/user/impl/UserServiceImpl.java 文件中添加禁用和启用账户的方法，具体代码如下。

```
/**
 * 更新账户
 */
@Override
public int updateByPrimaryKeySelective(User record) {
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用


```
        return userMapper.updateByPrimaryKeySelective(record);
    }
```

③ 禁用和启用账号数据层。

在 src/main/java/com/douxue/live/dao/mapper/UserMapper.java 文件中添加更新用户信息接口的代码，实现代码如下。

```
    int updateByPrimaryKeySelective(User record);
```

在 src/main/resources/mapper/UserMapper.xml 文件中添加更新用户信息动态 SQL，实现代码如下。

```
    <update id="updateByPrimaryKeySelective" parameterType="com.douxue.live.dao.entity.User">
        update t_user
        <set>
            <if test="wxUnionid != null">
                wx_unionid = #{wxUnionid,jdbcType=VARCHAR},
            </if>
            <if test="account != null">
                account = #{account,jdbcType=VARCHAR},
            </if>
            <if test="password != null">
                password = #{password,jdbcType=VARCHAR},
            </if>
            <if test="accountType != null">
                account_type = #{accountType,jdbcType=VARCHAR},
            </if>
            <if test="phone != null">
                phone = #{phone,jdbcType=VARCHAR},
            </if>
            <if test="headPic != null">
                head_pic = #{headPic,jdbcType=VARCHAR},
            </if>
            <if test="userName != null">
                user_name = #{userName,jdbcType=VARCHAR},
            </if>
            <if test="nickName != null">
                nick_name = #{nickName,jdbcType=VARCHAR},
            </if>
            <if test="state != null">
                state = #{state,jdbcType=INTEGER},
            </if>
            <if test="createTime != null">
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用


```
            create_time = #{createTime,jdbcType=TIMESTAMP},
        </if>
    </set>
    where user_id = #{userId,jdbcType=VARCHAR}
</update>
```

④ 数据交互。

在上一小节中，已经完成了禁启用账户的后端方法开发，接下来就要与后端进行数据交互，本任务采用 Ajax 与 JSON 来实现数据交互。

在界面代码 muser.jsp 中，交互代码如下所示。

```
// 启用
$('#data_table').on('click','.open',function(){
    var _self = $(this);
    layer.confirm('确定禁用该账户?',
        {
          btn: ['确定','取消'] //按钮
        }, function(){
            var id = _self.attr('data_id');
            $.post("<%= contextPath %>/web/admin/disaccount",{"userId":id, "state" : 0},
function(rs){
                if(rs.code == 1){
                    layer.msg("已启用",{icon:6});
                    $('#search_form').submit();
                } else {
                    layer.msg("失败",{icon:5});
                }
            });
    });
});
// 禁用
$('#data_table').on('click','.stop',function(){
    var _self = $(this);
    layer.confirm('确定禁用该账户?',
        {
          btn: ['确定','取消'] //按钮
        }, function(){
            var id = _self.attr('data_id');
            $.post("<%= contextPath %>/web/admin/disaccount",{"userId":id, "state" : -1},
function(rs){
                if(rs.code == 1){
                    layer.msg("已禁用",{icon:6});
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用


```
                $('#search_form').submit();
            } else {
                layer.msg("失败",{icon:5});
            }
        });
    });
});
```

⑤ 执行效果。

将项目部署到 Tomcat 中并发布，访问 http://localhost:8080/douxue_web/web/admin/home，在用户管理界面，单击“禁用”或“启用”按钮，禁用/启用账户，并在 APP 客户端进行登录测试，如图 7-13、图 7-14、图 7-15 和图 7-16 所示。

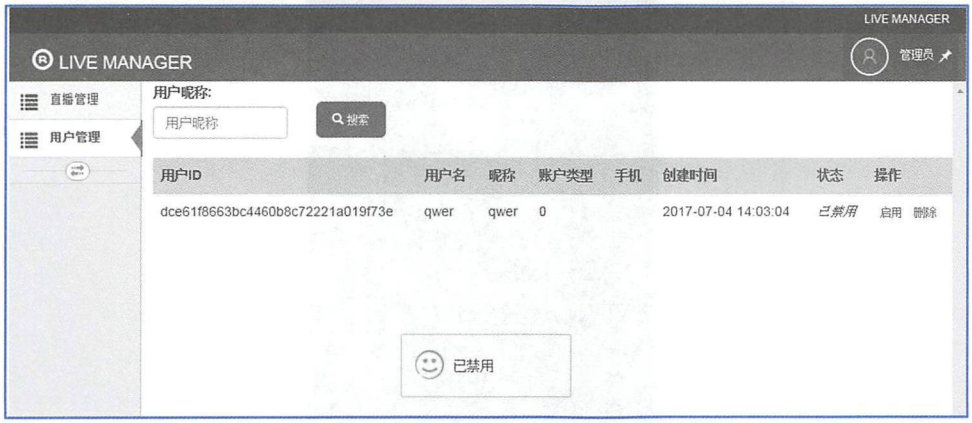


图 7-13
禁用账户

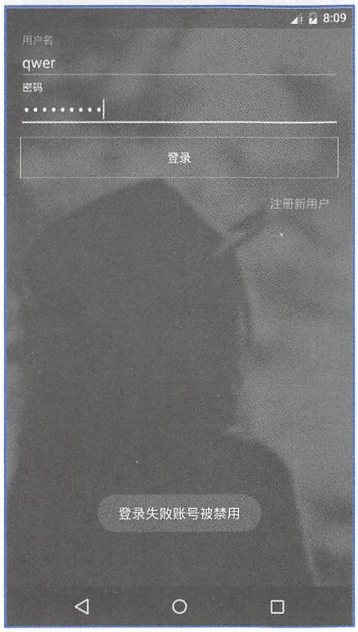


图 7-14
禁用账户登录

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

图 7-15
启用账户

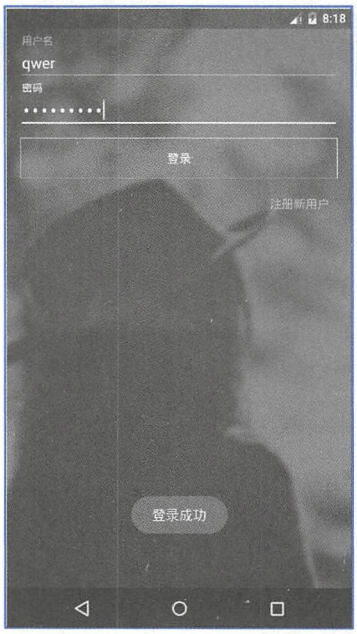


图 7-16
启用账户登录

3. 添加删除账户功能

① 删除用户控制层。

在 src/main/java/com/douxue/live/controller/web/admin/AdminController.java 文件中添加删除账户请求的方法，具体实现代码如下。

```
/**
 * 删除账户
 *
 * @param userId
 * @return
 */
@RequestMapping(value = "delaccount", method = RequestMethod.POST)
@ResponseBody
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用


```
public RestResult delAcccount(@RequestParam("userId") String userId) {
    int key = userService.deleteByPrimaryKey(userId);
    if (key > 0) {
        return RestResult.OK;
    }
    return RestResult.FAIL;
}
```

② 删除用户服务层。

在 src/main/java/com/douxue/live/service/user/impl/UserServiceImpl.java 文件中添加根据用户 ID 删除用户的方法，代码如下。

```
/**
 * 删除账户
 *
 * @param userId
 * @return
 */
@RequestMapping(value = "delaccount", method = RequestMethod.POST)
@ResponseBody
public RestResult delAcccount(@RequestParam("userId") String userId) {
    int key = userService.deleteByPrimaryKey(userId);
    if (key > 0) {
        return RestResult.OK;
    }
    return RestResult.FAIL;
}
```

③ 删除用户数据层。

在 src/main/java/com/douxue/live/dao/mapper/UserMapper.java 文件中添加删除用户接口的代码。

```
int deleteByPrimaryKey(String userId);
```

在 src/main/resources/mapper/UserMapper.xml 文件中添加删除用户动态 SQL，实现代码如下。

```
<delete id="deleteByPrimaryKey" parameterType="java.lang.String">
    delete from t_user
    where user_id = #{userId,jdbcType=VARCHAR}
</delete>
```

④ 数据交互。

在上一小节中，已经完成了删除账户的后端方法开发，接下来就要与后端进行数据

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

交互，本任务采用 Ajax 与 JSON 来实现数据交互。

在界面代码 muser.jsp 中，交互代码如下所示。

```
// 删除
$('#data_table').on('click','.delete',function(){
    var _self = $(this);
    layer.confirm('确定删除该账户?',
        {
          btn: ['确定','取消'] //按钮
        }, function(){
            var id = _self.attr('data_id');
            $.post("<%= contextPath %>/web/admin/delaccount",{"userId":id},function(rs){
                if(rs.code == 1){
                    layer.msg("已删除",{icon:6});
                    $('#search_form').submit();
                } else {
                    layer.msg("失败",{icon:5});
                }
            });
        });
    window.open("<%= contextPath %>/web/admin/delaccount?userId=" + id);
});
```

⑤ 执行效果。

将项目部署到 Tomcat 中并发布，访问 http://localhost:8080/douxue_web/web/admin/home，在用户管理界面，单击“删除”按钮，如图 7-17 和图 7-18 所示。

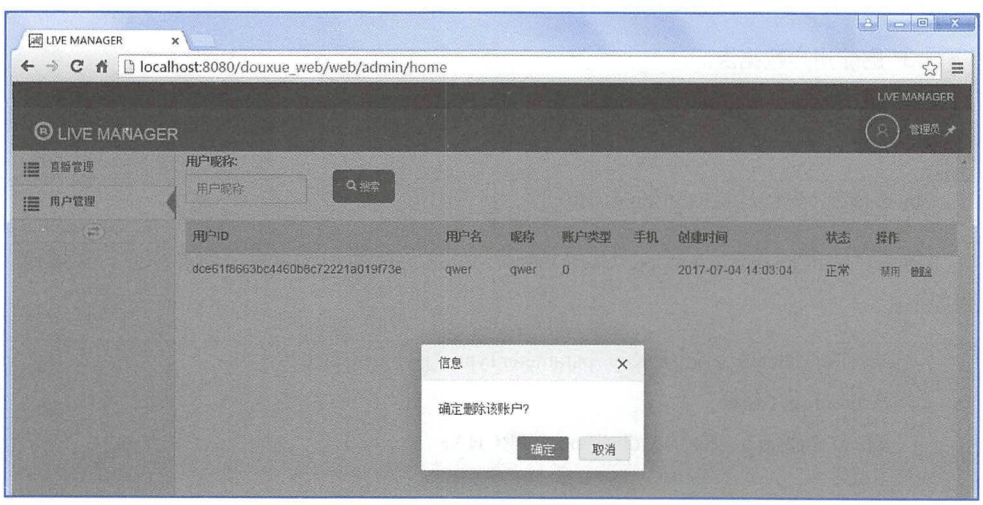


图 7-17
删除账户

版权所有，严禁传播，违者自负法律责任!

非卖品，仅供非商业用途或交流学习使用

图 7-18
删除账户成功

项目实训

实训题目

练习用户管理模块的开发。

实训目的

掌握 MyBatis 动态 SQL 的运用。

实训内容

开发一个 Demo，实现对系统用户的统一管理。

单元小结

本单元完成对直播和用户管理的部分功能，在后台可以实时监控视频直播画面，发现敏感内容时，可以通过禁播功能停止该直播间的视频直播，对于严重违规的用户，可以采取封禁账户的措施限制其登录直播客户端。

非卖品，仅供非商业用途或交流学习使用

单元 8 应用发布

学习目标

PPT8:
应用发布

知识目标

- 了解 Linux 的基本操作。
- 了解 APK 签名的作用。
- 了解 APK 签名的原理。

技能目标

- 掌握 Web 项目的部署方法。
- 掌握 Android 项目的签名打包方法。
- 掌握整套系统的执行测试方法。

非卖品，仅供非商业用途或交流学习使用

学习情境

斗学直播 APP 在小赵的努力下已经完成了整套系统的开发，现在需要将整套系统推出，小赵决定在腾讯云服务器上部署服务，同时用 Android Studio 对 APP 进行签名打包。

任务 8.1　部署 Web 项目

任务描述

初步了解 Linux 系统，熟悉 Linux 系统的目录结构及常用命令，并将 Web 系统部署在 Linux 服务器中，实现在网站的发布。

知识学习

1. Linux 简介

Linux 是一个自由的、免费的、源码开放的操作系统，也是开源软件中最著名的例子，其最主要是为了建立不受任何商品化软件版权制约的，全世界都能使用的类 UNIX 兼容产品。而将服务器部署在 Linux 上将会更加稳定、安全、高效及性能出色，这是 Windows 无法比的。

2. Linux 目录结构

Linux 目录结构如图 8-1 所示，具体说明见表 8-1。

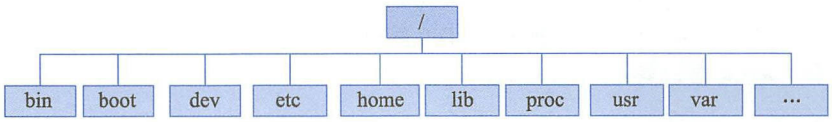


图 8-1
Linux 目录结构

表 8-1
Linux 目录结构说明

目　录	说　明
bin	存放二进制可执行文件（如 ls、cat、mkdir 等）
boot	存放用于系统引导时使用的各种文件
dev	用于存放设备文件
etc	存放系统配置文件
home	存放所有用户文件的根目录
lib	存放根文件系统中的程序运行所需要的共享库及内核模块
mnt	系统管理员安装临时文件系统的安装点
opt	额外安装的可选应用程序包所放置的位置
proc	虚拟文件系统，存放当前内存的映射
root	超级用户目录
sbin	存放二进制可执行文件，只有 root 才能访问
tmp	用于存放各种临时文件
usr	用于存放系统应用程序比较重要的目录/usr/local
var	用于存放运行时需要改变数据的文件

非卖品，仅供非商业用途或交流学习使用

任务实现

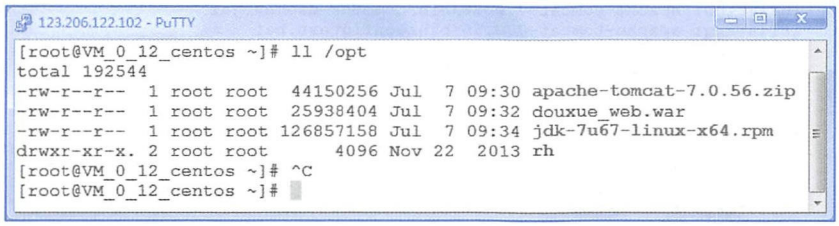
微课 8-1
部署 Web 项目

将本地 Web 工程部署到腾讯云服务器上，实现在公网可以访问直播服务。

① 通过 PuTTY 连接到云服务器，并将所需要的 jdk-7u67-linux-x64.rpm、apache-tomcat-7.0.56.zip、douxue_web.war 这 3 个软件包上传至自己的云服务器内，如图 8-2 所示。

工具 8-1：
PuTTY、jdk-7u67-linux-x64.rpm、pache-tomcat-7.0.56.zip、douxue_web.war

```
[root@VM_0_12_centos ~]# ll /opt
total 192544
-rw-r--r--  1 root root  44150256 Jul  7 09:30 apache-tomcat-7.0.56.zip
-rw-r--r--  1 root root  25938404 Jul  7 09:32 douxue_web.war
-rw-r--r--  1 root root 126857158 Jul  7 09:34 jdk-7u67-linux-x64.rpm
drwxr-xr-x. 2 root root      4096 Nov 22  2013 rh
```



源码 8-1：
部署 Web 项目

图 8-2
上传软件包

② 资源上传完毕后开始安装 JDK，如图 8-3 所示。

```
[root@VM_0_12_centos ~]# cd /opt
[root@VM_0_12_centos opt]# rpm -ivh jdk-7u67-linux-x64.rpm
Preparing...                ########################################### [100%]
   1:jdk                    ########################################### [100%]
Unpacking JAR files...
        rt.jar...
        jsse.jar...
        charsets.jar...
        tools.jar...
        localedata.jar...
        jfxrt.jar...
```

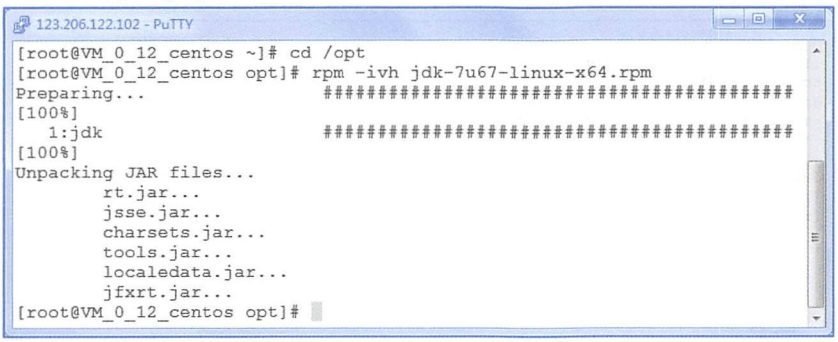


图 8-3
安装 JDK

③ JDK 安装完毕后，开始解压 Tomcat 服务器，如图 8-4 所示。

```
[root@VM_0_12_centos opt]# unzip apache-tomcat-7.0.56.zip
Archive:  apache-tomcat-7.0.56.zip
   creating: apache-tomcat-7.0.56/
   ......
```

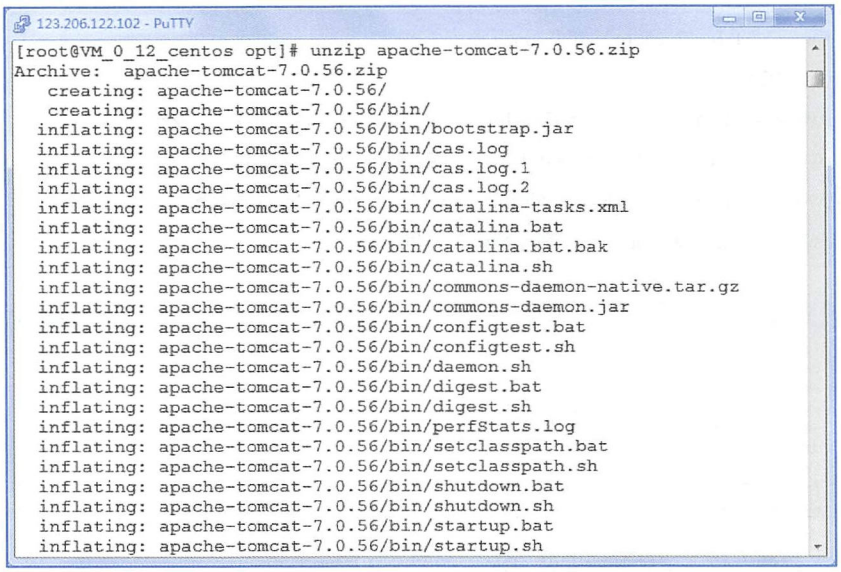


图 8-4
解压 Tomcat

④ Tomcat 解压完毕后，将 douxue_web.war 软件包复制到 Tomcat 的 webapps 目录下，如图 8-5 所示。

```
[root@VM_0_12_centos opt]# ll
total 192548
drwxr-xr-x  9 root root      4096 Oct 22  2014 apache-tomcat-7.0.56
-rw-r--r--  1 root root  44150256 Jul  7 09:30 apache-tomcat-7.0.56.zip
-rw-r--r--  1 root root  25938404 Jul  7 09:32 douxue_web.war
-rw-r--r--  1 root root 126857158 Jul  7 09:34 jdk-7u67-linux-x64.rpm
drwxr-xr-x. 2 root root      4096 Nov 22  2013 rh
[root@VM_0_12_centos opt]# cp douxue_web.war apache-tomcat-7.0.56/webapps/
[root@VM_0_12_centos opt]#
```

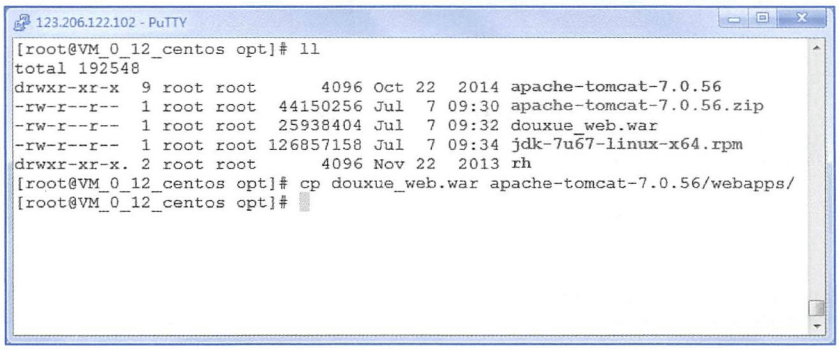


图 8-5
部署 WAR 包

⑤ 登录腾讯云账户，在导航栏中选择“云产品→数据库→关系型数据库”菜单命令，进入 MySQL 实例列表，选择一个实例进行登录（所选的云数据库必须能在部署 Web 项目的云主机内访问），如图 8-6 所示。

图 8-6
关系型
数据库

⑥ 输入账号、密码进行登录，在“数据管理”页面中，单击导航栏中的“返回 PMA”按钮，进入创建数据库页面，创建数据库 douxue_live，如图 8-7 所示。

图 8-7
登录数据库

⑦ 选择创建的数据库 douxue_live，导入 douxue_live.sql 文件构建数据库，如图 8-8 所示。

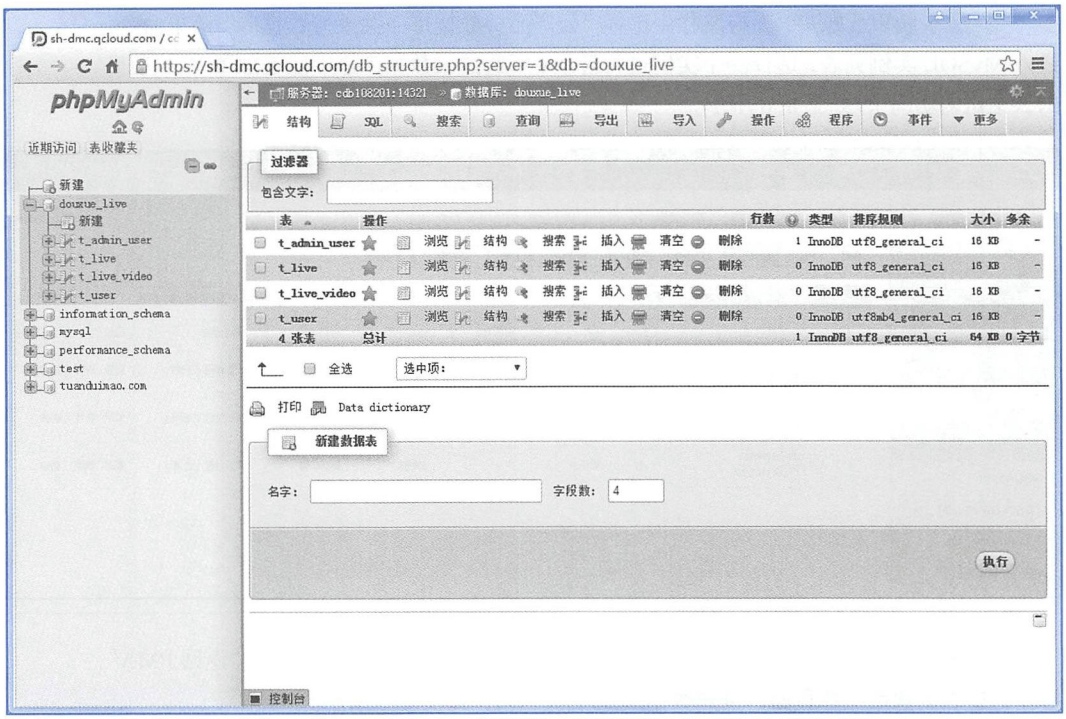


图 8-8
构建数据库

⑧ 数据库构建完成后，进入 Tomcat 服务器的 bin 目录内，给目录内的 shell 文件赋予可执行权限，准备启动服务器，如图 8-9 所示。

```
[root@VM_0_12_centos ~]# cd /opt/apache-tomcat-7.0.56/bin/
[root@VM_0_12_centos bin]# chmod +x *.sh
```

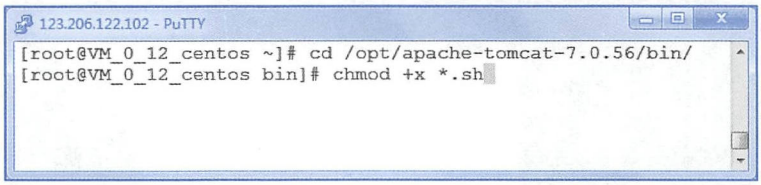


图 8-9
添加可执行条件

⑨ 启动 Tomcat 服务器，如图 8-10 所示。

```
[root@VM_0_12_centos bin]# ./startup.sh
Using CATALINA_BASE:   /opt/apache-tomcat-7.0.56
Using CATALINA_HOME:   /opt/apache-tomcat-7.0.56
Using CATALINA_TMPDIR: /opt/apache-tomcat-7.0.56/temp
Using JRE_HOME:        /usr
Using  CLASSPATH:      /opt/apache-tomcat-7.0.56/bin/bootstrap.jar:/opt/apache-tomcat-
7.0.56/ bin/tomcat-juli.jar
Tomcat started.
[root@VM_0_12_centos bin]#
```

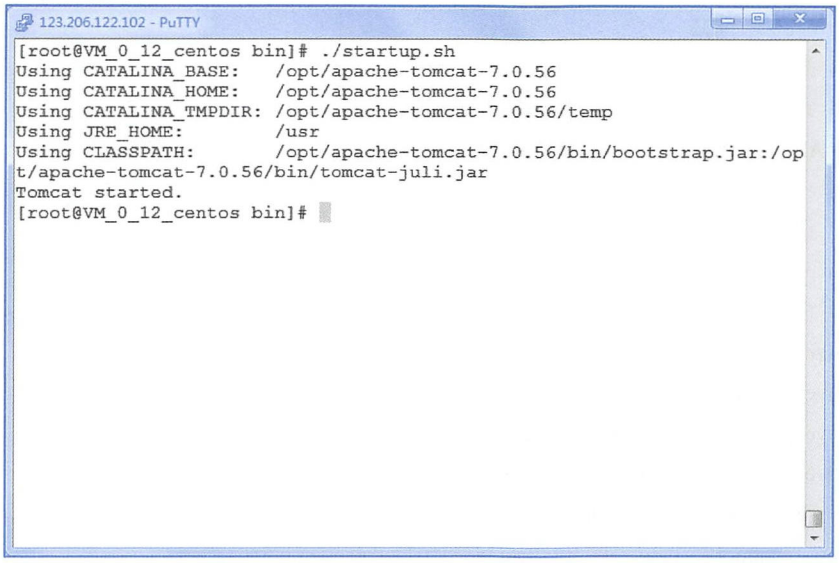


图 8-10
启动 Tomcat

⑩ 进入 Tomcat 服务器的 webapps/douxue_web/WEB-INF/classes 目录内修改 application.properties 文件，如图 8-11 所示。将申请的各项服务参数配置正确，具体如下代码所示。

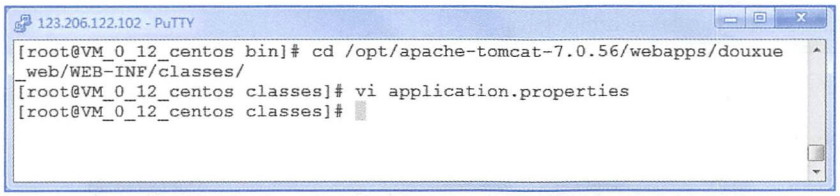


图 8-11
配置服务参数

```
# Mysql
jdbc_url=jdbc:mysql://192.168.0.8:3306/douxue_live
jdbc_username=root
jdbc_password=xiandian55

# Redis
redis_uri=10.66.239.188
redis_port=6379
redis_password=crs-fjp8n6fs:xiandian55
redis_timeout=2000
redis_pool_maxIdle=100
redis_pool_maxTotal=1000

# 直播相关配置
# APP_ID
live_appid=1253440178
# BIZID
```

```
live_bizid=9835
# 推流防盗链 KEY
live_push_safe_key=a32bf57f0a91e26536a563e492be9c86
# API 鉴权 KEY
live_api_sign_key=649b081815bf965e0f86dc125dc895f2
# 直播回放切片生成格式
live_record=mp4
# 直播相关推/拉流地址
live_push_rtmp_url=rtmp://%s.livepush.myqcloud.com/live/%s?bizid=%s
live_play_rtmp_url=rtmp://%s.liveplay.myqcloud.com/live/%s
live_play_flv_url=http://%s.liveplay.myqcloud.com/live/%s.flv
live_play_hls_url=http://%s.liveplay.myqcloud.com/live/%s.m3u8
# 直播公共相关 API 地址
live_common_access_url=http://fcgi.video.qcloud.com/common_access
# 腾讯云存储
qcloud_cos_secret_id=AKIDKXZP3cDDOQHyTlqBZqwqY5RbaNyNunxx
qcloud_cos_secret_key=0oWFn9KK8kAOQGuNdtzvfCXhfT8bszuf
qcloud_cos_app_id=1253440178
qcloud_cos_bucket=xiaozhibo4java
```

⑪ 确认配置正确后，通过浏览器访问“http://本机服务器地址:服务端口/douxue_web”，如图 8-12 所示。

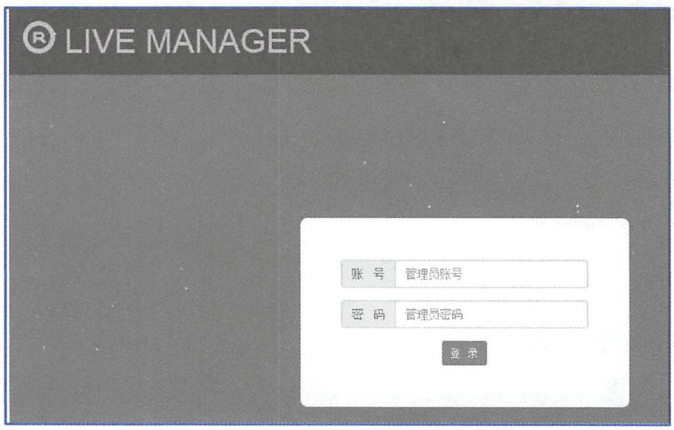


图 8-12
运行效果

项目实训

实训题目

练习 Web 系统的部署。

实训目的

1. 了解 Linux 系统的目录结构和常用命令。

非卖品，仅供非商业用途或交流学习使用

2. 掌握在 Linux 系统中部署 Web 系统的方法。

实训内容

部署 Web 系统，实现网站的发布。

任务 8.2　签 名 打 包

任务描述

笔 记

了解 APK 签名的作用及其原理后，对 Android 项目进行签名打包，完成后安装打包出来的 APK，配合后台执行测试。

知识学习

1. APK 签名的作用

在 Android 系统中，所有安装到系统的应用程序都必有一个数字证书，此数字证书用于标识应用程序的作者和在应用程序之间建立信任关系，如果一个 permission 的 protectionLevel 为 signature，那么就只有那些跟该 permission 所在的程序拥有同一个数字证书的应用程序才能取得该权限。Android 使用 Java 的数字证书相关的机制来给 APK 加盖数字证书，要理解 Android 的数字证书，需要先了解以下数字证书的概念和 Java 的数字证书机制。Android 系统要求每一个安装进系统的应用程序都是经过数字证书签名的，数字证书的私钥则保存在程序开发者的手中。Android 将数字证书用来标识应用程序的作者和在应用程序之间建立信任关系，不是用来决定最终用户可以安装哪些应用程序。这个数字证书并不需要权威的数字证书签名机构认证，它只是用来让应用程序包自我认证的；也是用来判断该应用是否被别人破解，是二次打包的一个标准，但是签名并不能防止被破解。

Android 数字证书包含以下几个要点。

① 所有的应用程序都必须有数字证书，Android 系统不会安装一个没有数字证书的应用程序。

② Android 程序包使用的数字证书可以是自签名的，不需要一个权威的数字证书机构签名认证。

③ 如果要正式发布一个 Android，必须使用一个合适的私钥生成的数字证书来给程序签名，而不能使用 ADT 插件 或者 ant 工具 生成的调试证书来发布。

④ 数字证书都是有有效期 的，Android 只是在应用程序安装的时候才会检查证书的有效期。如果程序已经安装在系统中，即使证书过期也不会影响程序的正常功能。

⑤ Android 使用标准的 Java 工具 Keytool and Jarsigner 来生成数字证书，并给应用程序包签名。

⑥ 使用 zipalign 优化程序。

2. 签名原理

Android 系统签名主要有 ROM 签名和应用程序 APK 签名两种形式。ROM 签名是针

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

对已经生成的 Android 系统 ROM 包进行签名。应用程序 APK 签名是针对开发者开发的应用程序安装包 APK 进行签名的。前者是对整个 Android 系统包签名，后者只对 Android 系统中一个应用程序 APK 签名。

Android 应用程序 APK 是 jar 包，签名采用的工具是 signapk.jar 包，对应用程序安装包签名的执行命令如下。

```
java-jar signapk.jar publickey privatekey input.apk output.apk
```

此命令实现了对应用程序安装包 input.apk 签名的功能。在 signapk.jar 命令中，第 1 个参数为公钥 publickey，第 2 个参数为私钥 privatekey，第 3 个参数为输入的包名，第 4 个参数为签名后生成的输出包名。在此命令中，signapk.jar 使用公钥 publickey 和私钥 privatekey 对 input.apk 安装包进行签名，生成 output.apk 包。signapk 源码位于 build/tools/signapk/SignApk.java 中。

完成签名后 APK 包中多了一个 META-INF 文件夹，其中有名为 MANIFEST.MF、CERT.SF 和 CERT.RSA 的 3 个文件。MANIFEST.MF 文件中包含很多 APK 包信息，如 manifest 文件版本、签名版本、应用程序相关属性、签名相关属性等。CERT.SF 是明文的签名证书，通过采用私钥进行签名得到。CERT.RSA 是密文的签名证书，通过公钥生成的。MANIFEST.MF、CERT.SF 和 CERT.RSA 这 3 个文件所使用的公钥和私钥的生成可以通过 development/tools/make_key 来获得。下面分别介绍 MANIFEST.MF、CERT.SF 和 CERT.RSA 这 3 个文件的生成方法。

（1）生成 MANIFEST.MF 文件

生成 MANIFEST.MF 文件是对 APK 包中所有未签名文件逐个用算法 SHA1 进行数字签名，再对数字签名信息采用 Base64 进行编码，最后将编完码的签名写入 MANIFEST.MF 文件中的。添加数字签名到 manifest 文件通过调用 addDigestsToManifest 方法实现，具体代码如下（简而言之，MANIFESt.MF 文件中的内容就是通过遍历 APK 中所有文件（entry），逐一生成 SHA-1 数字签名，然后通过 Base64 编码转码的）。

```
{ Private static Manifest addDigestsToManifest(JarFile.jar)
…
//遍历 update.apk 包中所有文件
//得到签名文件内容
InputStream data=jar.getInputStream(entry);
//更新文件内容
while((num=data.read(buffer))>0){
md.update(buffer, 0, num);
}
…
//进行 SHA1 签名，并采用 Base64 进行编码
attr.putValue("SHA1-Digest",base64.encode(md.digest()));
output.getEntries().put(name,attr);
…
}
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

说明

生成 MANIFEST.MF 使用了 SHA1 算法进行数字签名，SHA1 是一种 Hash 算法，两个不同的信息经 Hash 运算后不会产生同样的信息摘要，由于 SHA1 是单向的，所以不可能从消息摘要中复原原文。如果恶意程序改变了 APK 包中的文件，那么在进行 APK 安装校验时，改变后的摘要信息与 MANIFEST.MF 的检验信息不同，应用程序便不能安装成功。

（2）生成 CERT.SF 文件

在生成 MANIFEST.MF 文件之后，用 SHA1-RSA 算法对其进行私钥签名（使用 SHA1-RSA 算法，用私钥对 MANIFEST.MF 摘要文件签名，并对其中每个文件摘要签名），便生成 CERT.SF。具体代码如下。

```
Signature signature = Signature.getInstance("SHA1withRSA");
signature.initSign(privateKey);
je=newJarEntry(CERT_SF_NAME);
je.setTime(timestamp);
outputJar.putNextEntry(je);
writeSignatureFile(manifest,newSignatureOutputStream(outputJar,signature));
```

RSA 是目前最有影响力的公钥加密算法，是一种非对称加密算法，能够同时用于加密和数字签名。由于 RSA 是非对称加密算法，因此用私钥对生成 MANIFEST.MF 的数字签名加密后，在 APK 安装时只能使用公钥才能解密它。

（3）生成 CERT.RSA 文件

生成 CERT.RSA 文件与生成 CERT.SF 文件的不同之处在于，生成 CERT.RSA 文件使用了公钥文件。CERT.RSA 文件中保存了公钥及所用的采用加密算法等信息。具体代码如下。

```
Je =new JarEntry(CERT_RSA_NAME);
je.setTime(timestamp);
outputJar.putNextEntry(je);
writeSignatureBlock(signature,publicKey,outputJar);
```

通过以上对 Android 应用程序签名的代码分析，可以看出 Android 系统通过对第三方 APK 包进行签名，达到保护系统安全的目的。应用程序签名主要用于对开发者的身份进行识别，达到防范恶意攻击的目的，但不能有效地限制应用程序被恶意修改，只能够检测应用程序是否被修改过，如果应用程序被修改应该再采取相应的应对措施。

微课 8-2
签名打包

任务实现

1. Android 项目签名打包

源码 8-2：
签名打包

① 根据自己的腾讯云服务配置 src\main\java\com\tencent\qcloud\gogal\base\TCConstants.java 文件中的各项参数，如图 8-13 所示。

```
/**
* 腾讯云服务配置专区（请自主替换）发布使用
```


版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用


```
*/
//云通信服务相关配置
public static final int IMSDK_ACCOUNT_TYPE = 13703;
public static final int IMSDK_APPID = 1400033541;

//COS 存储服务相关配置
public static final String COS_BUCKET = "xiaozhibo4java";
public static final String COS_APPID   = "1253440178 ";

//业务 Server 的 HTTP 配置
public static final String SVR_POST_URL= "http://本机服务器地址:服务端口/douxue_web/app/";
```

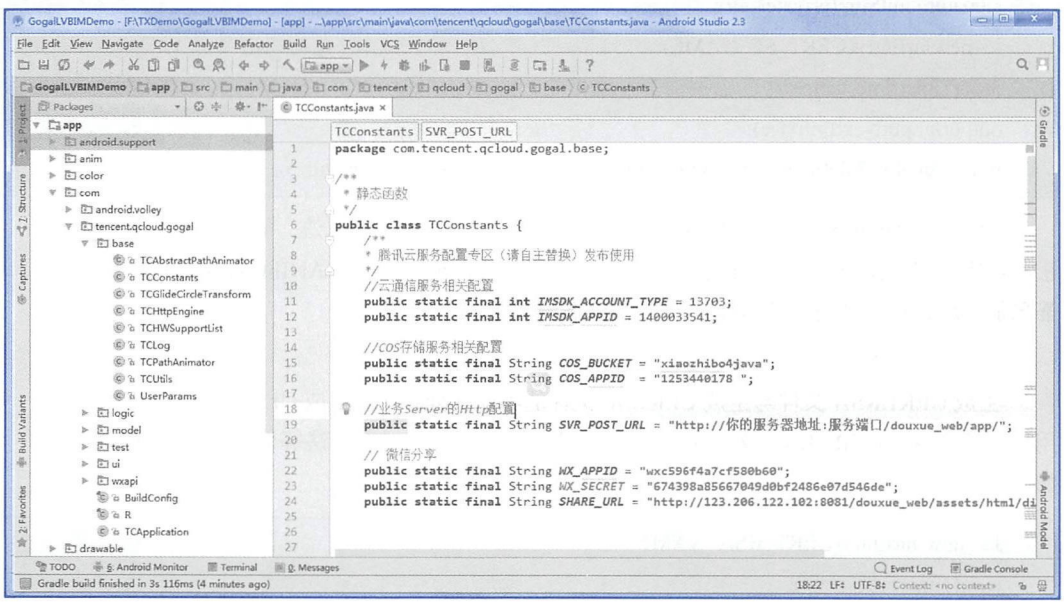


图 8-13
参数配置

② 配置完成后，选择“Build→Generate Signed APK”菜单命令，如图 8-14 所示。

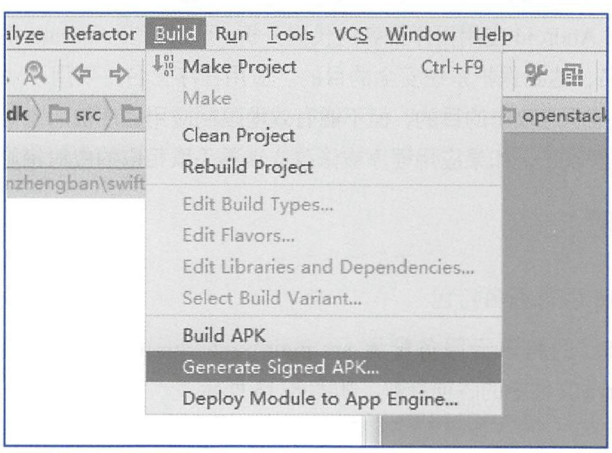


图 8-14
选择生成签名 APK 命令

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

③ 在弹出窗口中单击“Create new”按钮来创建 Key，如图 8-15 所示。

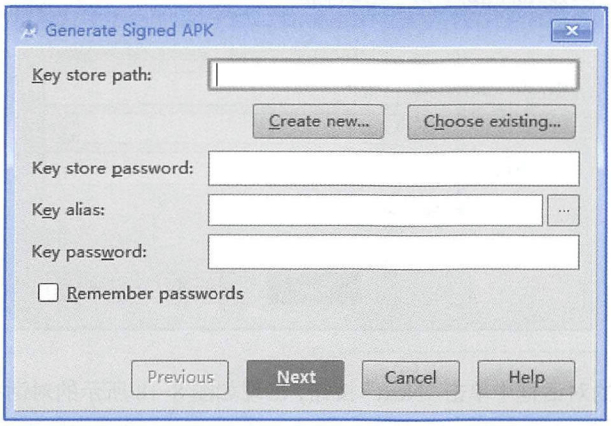


图 8-15
创建新 Key

④ 在弹出窗口中设置对应的属性。其中，“Key store path”文本框用来输入产生 Key 文件的保存路径，“Key store password”文本框用来输入 Key 的存储密码，“Key alias”文本框用来输入 Key 的别名，“Key password”文本框用来输入 Key 的密码，如图 8-16 所示。

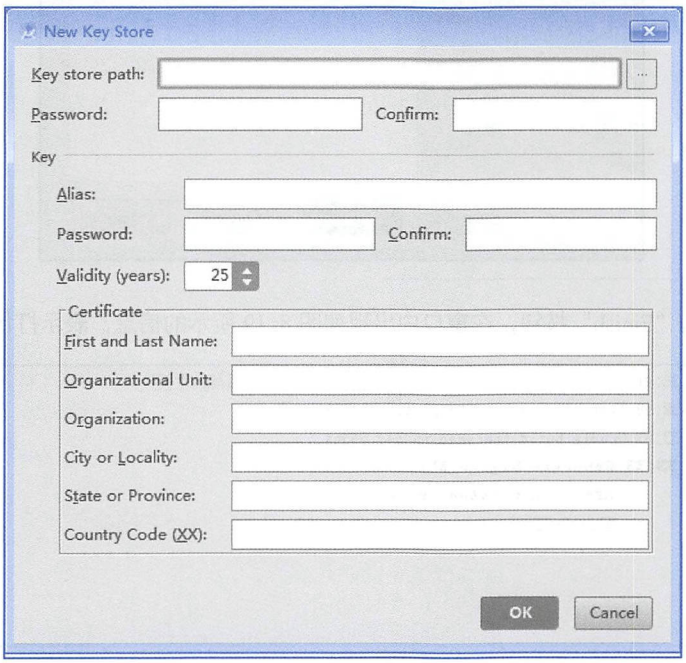


图 8-16
设置 Key 相关属性

⑤ 输入完成后单击“OK”按钮，跳转到相应的对话框，单击“Next”按钮进入下一步，如图 8-17 所示。

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

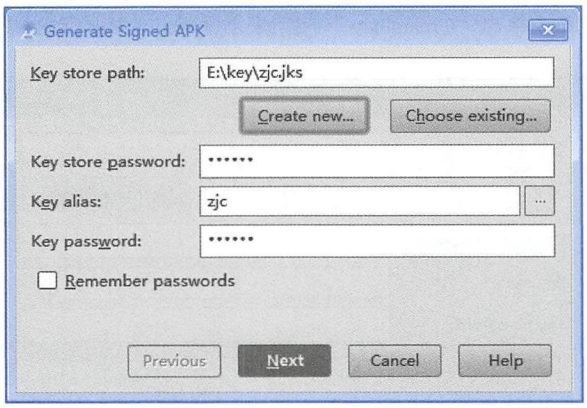


图 8-17
属性设置完成

⑥ 在弹出的对话框中单击“Next”按钮，出现如图 8-18 所示的对话框。其中，“APK Destination Folder”表示选择 APK 的保存路径，单击“倒三角”按钮▼，在“Build Type”下拉列表中选择“release”选项。

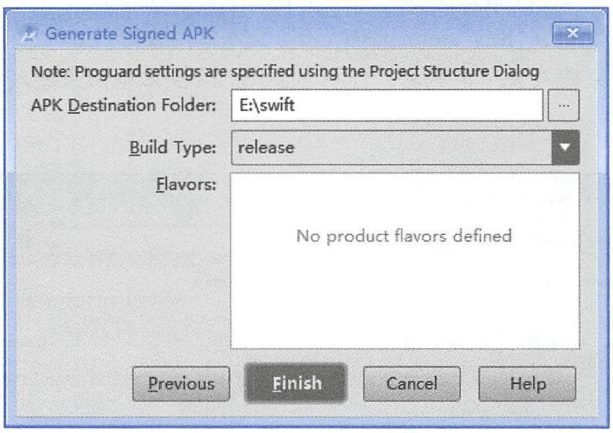


图 8-18
选择发布结果选项

⑦ 单击“Finish”按钮，在窗口中出现如图 8-19 所示的信息，表示打包成功。

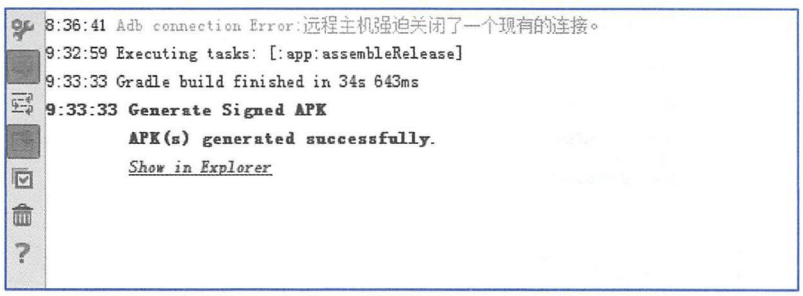


图 8-19
APK 生成成功

2. 执行测试

① 使用 Android 手机安装打包好的 APK 安装包，启动应用并登录，如图 8-20 所示。

② 单击按钮开启新的直播，并测试直播界面包含的功能，如图 8-21 所示。

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

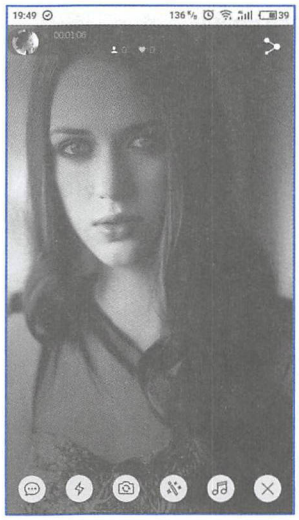


图 8-20
用户登录

图 8-21
APP 直播

③ 通过浏览器访问“http://本机服务器地址:服务端口/douxue_web”并登录系统，查看直播列表，如图 8-22 所示。

图 8-22
直播列表

单击“监控”按钮，查看当前直播间的直播内容，如图 8-23 所示。

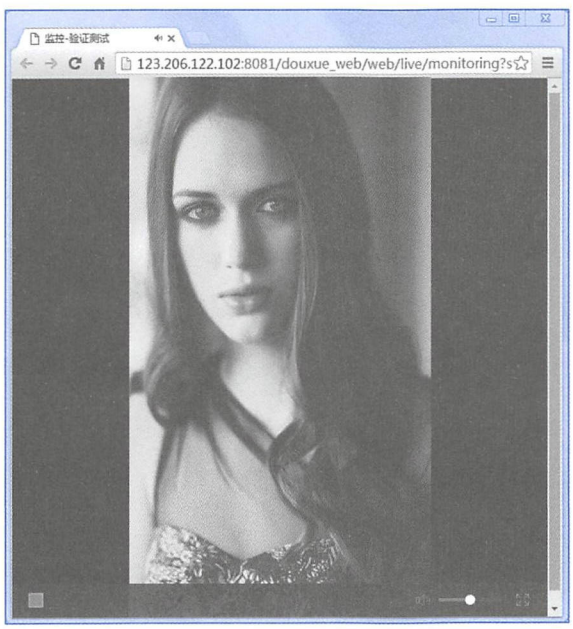


图 8-23
监控直播

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

项目实训

实训题目

练习签名打包。

实训目的

熟悉并掌握签名打包的过程。

实训内容

新建一个项目进行签名打包。

单元小结

本单元完成了整个项目中 Web 后台的项目部署及 Android 项目的签名打包。在 Android 项目打包完成后进行了前后台的安装验证，至此整套简单的直播系统就完成了。

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

附录

iOS 和小程序直播

腾讯移动直播提供了 iOS 直播、小程序直播开放的案例，可以到如下地址进行下载和学习：https://cloud.tencent.com/document/product/454/6991，具体如表 A-1 所示。

表 A-1
案例资源包及说明

类型	版本号	说明
iOS 源码包	2.0.4	由 RTMP SDK、IM SDK、COS SDK、业务逻辑层代码和界面层代码构成
Android 源码包	2.0.4	由 RTMP SDK、IM SDK、COS SDK、业务逻辑层代码和界面层代码构成
PHP 源码包	2.0.3	为观众端提供直播间列表，同时为主播端提供推流 URL
Web 分享页	1.1.0	基于 HTML 5 技术构建的 Web 分享页面，支持在常规手机浏览器和 PC 浏览器上观看直播，并支持与主播进行消息互动
微信小程序	1.1.0	在（微信→发现→小程序）中搜索“小直播”可以体验效果，目前微信还不支持直播推流能力，只能用来观看视频直播和点播

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

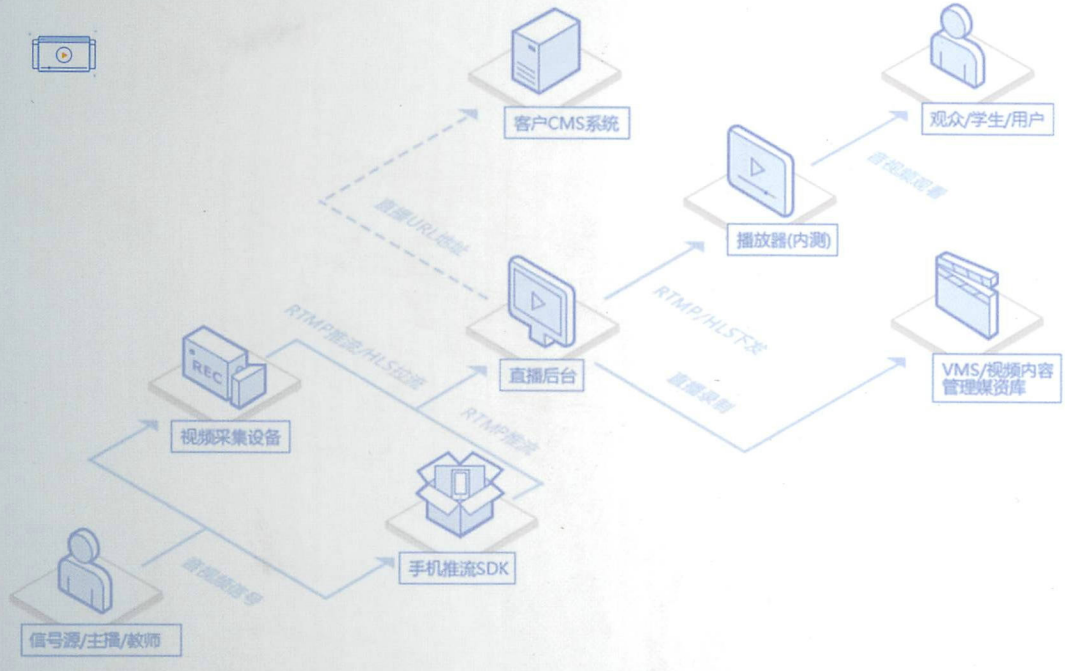
参考文献

[1] 腾讯云官网，https://www.qcloud.com/.
[2] 杜纪魁，沈建国. Android 云存储客户端开发. 北京：高等教育出版社，2017.
[3] 张桂珠，刘丽，陈爱国. Java 面向对象程序设计. 2 版. 北京：北京邮电大学出版社，2007.
[4] 陈衍席. 基于 Java 网络聊天系统的设计及实现研究. 通讯世界，2016.
[5] 叶晓静，黄俊伟. 基于 Android 系统的多媒体播放器解决方案. 现代电子技术，2011.

版权所有，严禁传播，违者自负法律责任！

CETC 云计算技术与应用专业校企合作系列教材

- 云计算综合运维管理
- 微信小程序应用开发
- 企业SaaS云应用开发
- 大数据分析应用开发
- 全景画面直播应用开发
- 人脸识别和美化应用开发
- 游戏应用开发
- 语音智能机器人应用开发
- AI博弈算法应用开发
- ✓ 视频直播APP应用开发



ISBN 978-7-04-049016-9
9 787040 490169 >
定价 38.50 元